DOI: 10.1111/tgis.12267

RESEARCH ARTICLE

WILEY Transactions in GIS

A spatial parallel heuristic approach for solving very large-scale vehicle routing problems

Wei Tu[1] | Qingquan Li[1] | Qiuping Li[2] | Jiasong Zhu[1] | Baoding Zhou[1] | Biyu Chen[3]

[1] Shenzhen University Shenzhen Key Laboratory of Spatial Smart Sensing and Services, College of Civil Engineering & Key Laboratory for Geo-Environmental Monitoring of Coastal Zone of the National Administration of Surveying, Mapping and GeoInformation, Shenzhen, Guangdong, China

[2] Sun Yat-Sen University, Center of Integrated Geographic Information Analysis, School of Geography and Planning, Guangzhou, Guangdong, China

[3] Wuhan University State Key Laboratory of Information Engineering in Surveying, Mapping, and Remote Sensing, Wuhan, Hubei, China

Correspondence
Qiuping Li, Sun Yat-Sen University, Center of Integrated Geographic Information Analysis, School of Geography and Planning, Guangzhou, Guangdong, China
Email: liqp3@mail.sysu.edu.cn

Jiasong Zhu, Shenzhen University Shenzhen Key Laboratory of Spatial Smart Sensing and Services, College of Civil Engineering & Key Laboratory for Geo-Environmental Monitoring of Coastal Zone of the National Administration of Surveying, Mapping and GeoInformation, Shenzhen, Guangdong, China
Email: zhujiasong@gmail.com

Abstract

The vehicle routing problem (VRP) is one of the most prominent problems in spatial optimization because of its broad applications in both the public and private sectors. This article presents a novel spatial parallel heuristic approach for solving large-scale VRPs with capacity constraints. A spatial partitioning strategy is devised to divide a region of interest into a set of small spatial cells to allow the use of a parallel local search with a spatial neighbor reduction strategy. An additional local search and perturbation mechanism around the border area of spatial cells is used to improve route segments across spatial cells to overcome the border effect. The results of one man-made VRP benchmark and three real-world super-large-scale VRP instances with tens of thousands of nodes verify that the presented spatial parallel heuristic approach achieves a comparable solution with much less computing time.

KEYWORDS

Local search, logistics, parallel computing, spatial partition, vehicle routing problems

1 | INTRODUCTION

The vehicle routing problem (VRP) is one of the most prominent problems in the literature of spatial optimization. A solution of a VRP aims to find optimal routes to serve a set of geographically scattered customers such that each customer is served once and the vehicle does not operate beyond its capacity. It has many real-world applications in both the public and private sectors, such as parcel delivery (Jung, Lee, & Chun, 2006), waste collection (Arribas, Blazquez, & Lamas, 2010; Xue & Cao, 2016), goods distribution (Fang, Tu, Li, Shaw, Chen, & Chen, 2013), courier delivery

(Janssens, Van den Berghm, Sörensen, & Cattrysse, 2015), etc. To handle the complex real-world applications, a large amount of research has targeted solving VRP variants, including vehicle routing problems with time windows (VRPTW) (Zhang, Lam, & Chen, 2013), vehicle routing problems with multiple use of vehicle (VRPM), vehicle routing problems with simultaneous pickup–delivery and time windows (VRPPDTW) (Wang, Mu, Zhao, & Sutherland, 2015), and vehicle routing problems with stochastic demand (VRPSD) (Zhang et al., 2016).

Another strand of research that has received much attention is the development of efficient solution algorithms for solving VRP as well as its variants (Weigel & Cao, 1999; Miller & Shaw, 2000; Mendoza, Medaglia, & Velasco, 2009; Kuo, Lord, & Walden, 2013; Tu, Fang, Li, Shaw, & Chen, 2014a; Tu, Li, Chang, Yue, & Zhu, 2015a; Tu, Li, Fang, & Zhou, 2015b; Kinobe, Bosona, Gebresenbet, Niwagaba, & Vinneras, 2015). Because the VRP and its variants cannot be solved exactly in nondeterministic polynomial time, most existing algorithms rely on effective heuristic approaches to find near-optimal solutions. Comprehensive reviews of the VRP solution algorithms can be found in Golden, Raghavan, and Wasil (2008), Laporte (2009), Vidal, Crainic, Gendreau, and Prins (2013), and Toth and Vigo (2014). State-of-the-art heuristic algorithms combined with spatial reduction strategies such as the nearest neighbor (Li, Golden, & Wasil, 2005), the granular neighbor (Toth & Vigo, 2003), the kth-ring Voronoi neighbor (Fang et al., 2013), and spatial clustering (Qi, Lin, Li, & Miao, 2012), can provide a promising solution in several hours for VRPs with thousands of customers. Although these state-of-the-art algorithms have made significant contributions to the VRP applications, more efficient algorithms are obviously needed to support online vehicle dispatching in large-scale real-world applications (Qi et al., 2012; Curtin, Voicu, Rice, & Stefanidis, 2014; Li, Chen, Wang, & Lam, 2015).

Parallel computing, which utilizes multiple computing units, has been proven to be an effective way to address computation-intensive problems (Qin, Zhan, Zhu, & Zhou, 2015), such as forest fire spread forecasting and path planning (Djidjev, Chapuis, Andonov, Thulasidasan, & Lavenier, 2015; Li, Cao, & Church, 2016; Artés, Cencerrado, Cortés, & Margalef, 2016). This study aims to develop an efficient VRP solution algorithm based on parallel computing techniques.

In the literature, relatively little attention has been given to utilizing parallel computing techniques for developing efficient VRP algorithms. The major obstacle to using parallel computing techniques in VRPs is related to the decomposition mechanism that divides the complex spatial optimization problem into a set of sub-problems for parallel computing. The concurrent search mechanism is often used for decomposition, where multiple parallelized processes concurrently search the entire solution and communicate with one another for valuable information (Polacek, Benkner, Doerner, & Hartl, 2008; Groër, Golden, & Wasil, 2011; Jin, Crainic, & Lokketangen, 2012; Wang et al., 2015). One popular concurrent search mechanism is to explore the solution space with different parameters, i.e. parallel iterated local search with different parameters for VPR with simultaneous pickup and delivery (Subramanian, Drummond, Bentes, Ochi, & Farias, 2010), cooperative parallel searching with a solution pool (Jin, Crainic, & Løkketangen, 2014). However, the concurrent optimization of a set of VRP solutions is not easy to achieve because of possible node interdependence such that multiple searching processes may evaluate the same node (Vidal et al., 2013).

An alternative mechanism is the spatial decomposition mechanism, which divides the region of interest into a set of small spatial cells and then derives partial VRP solutions in these spatial cells, coordinated by multiple computing cores. This spatial decomposition mechanism has been verified to be effective for sequential computing. Ouyang (2007) developed a non-overlapping disk model for solving large-scale VRPs with 2,000 customers. Qi et al. (2012) proposed a spatiotemporal clustering-based partitioning method to assist in the construction of the initial solution for a large-scale VRP with time windows. However, both of them are embedded within the sequential searching heuristic. The combination of a concurrent search mechanism and a spatial decomposition mechanism for a large scale VRP has still not been well investigated.

This article presents a novel spatial parallel heuristic approach for solving very large vehicle routing problems with more than 10,000 customers. As opposed to the traditional decomposition mechanism that focuses on the solution space, from the spatial view it decomposes the VRP region into many small spatial cells and concurrently improves the partial solution within each cell with a parallel optimization technique. Spatial partitioning is used to divide a large-scale VRP instance into many small sub-VRPs that can be concurrently improved with local search-based heuristics. Border areas surrounding the boundary of each partition will be identified to facilitate local search in these special areas to overcome the consequent boundary effect on the solution improvement phase. Intensive experiments on many real-

world large-scale VRP datasets have been conducted, and the results demonstrate that the proposed approach is able to report solutions to very large-scale VRP instances that are highly competitive with the best solutions in the literature and state-of-the-art heuristics.

The remainder of this paper is organized as follows. The next section provides a brief VRP definition. The principle of spatial parallel optimization is explored in Section 3. The spatial parallel solution approach is presented in Section 4. Section 5 reports the comparison between the approach presented in this article and the state-of-the-art heuristics. Section 6 discusses the speedup performance, the impact of perturbation size, and the trade-off between solution quality and computing effort. Finally, conclusions are drawn in Section 7.

2 | PROBLEM STATEMENT

VRP aims to design the best routes to serve many nodes with side constraints. Formally, let $G=(V,\ E)$ be a complete graph with a vertex set $V=\{v_1, v_2, \cdots, v_n, v_{n+1}\}$ and an edge set $E=\{(v_i, v_j)|\ v_i, v_j \in V, i \neq j,\ i,j \in N, 1 \leq i,\ j \leq n+1\}$. Vertices $V_c=\{v_1, v_2, \cdots, v_n\}$ denote customers. Each customer $v_i \in V_c$ has a nonnegative demand, q_i. The remaining vertex $V_D=\{v_{n+1}\}$ denotes the depot. Each edge $e_{ij}=(v_i, v_j)$ has a nonnegative travel distance/time, d_{ij}/t_{ij}, representing the travel cost from v_i to v_j. A fleet of K identical vehicles is located at the depot. Each vehicle departs from the depot, serves many customers, and returns to the depot.

A route $r=\{v_0, v_2, \cdots, v_R, v_{R+1}\}$ is a sequence of customers visited by a vehicle, where $v_0 \in V_D$, $v_{R+1}=v_0$. The total served demand $\sum_{i=1}^{R} q_{v_i}$ of the route should be no more than the vehicle capacity Q. A route is defined as feasible if the vehicle capacity is satisfied. The total travel distance of the route is defined as $\sum_{i=0}^{R} d_{v_i v_{i+1}}$. A solution s of the VRP is a set of feasible routes $\{r_1, r_2, \cdots, r_K\}$ such that each customer is visited once by a single vehicle. Following the ordinary assumption of VRP, Euclidean distance is used as travel cost. The objective is to minimize the total number of vehicles used and the total travel distance. It should be noted that network distance and travel time are alternative cost indexes.

3 | PRINCIPLE OF THE SPATIAL PARALLEL METAHEURISTICS FOR THE VRP

Before providing a description of the principle of the spatial parallel metaheuristics for the VRP, local search and related definitions regarding spatial partitioning of the VRP will be introduced.

3.1 | Local search

Local search is the most successful heuristic solution technique for solving the VRP (Laporte, 2009). It explores the solution space with small modifications to the current solution by moving or exchanging nodes within a route or between routes to improve the solution quality. The way to move or exchange nodes within a route or between routes is called a neighborhood structure (Zachariadis & Kiranoudis, 2010). Three common neighborhood structures are briefly introduced below.

- *Relocate*(*i*, *j*): A node v_i is moved from route r_1 to be before or after node v_j in route r_2. Figure 1a illustrates a relocation operation, in which node v_i is moved from its original place and inserted before node v_j in route r_2. The objective gain is $\Delta s=(d_{i-1,i+1}+d_{j-1,i}+d_{ij})-(d_{i-1,i}+d_{i,i+1}+d_{j-1j})$.
- *Swap*(*i*, *j*): Interchange node v_i from route r_1 with node v_j from route r_2. Figure 1b illustrates a swap operation, in which the positions of nodes v_i *and* v_j are exchanged. The objective gain is $\Delta s=(d_{i-1,j}+d_{j,i+1}+d_{j-1,i}+d_{ij+1})-(d_{i-1,i}+d_{i,i+1}+d_{j-1,j}+d_{jj+1})$.
- *SwapEnd*(*i*, *j*): Swaps the end of route r_1 after node v_i and the end of route r_2 after node v_j. Figure 1c gives an example of the SwapEnd operation, in which the partial routes after nodes v_i and v_j are exchanged. The objective gain is $\Delta s=(d_{ij+1}+d_{j,i+1})-(d_{i,i+1}+d_{jj+1})$.

The nature of local search may cause it to drop into local minima after a number of operations with neighborhood structures (Golden et al., 2008). Many effective local search strategies and algorithms have been introduced to do

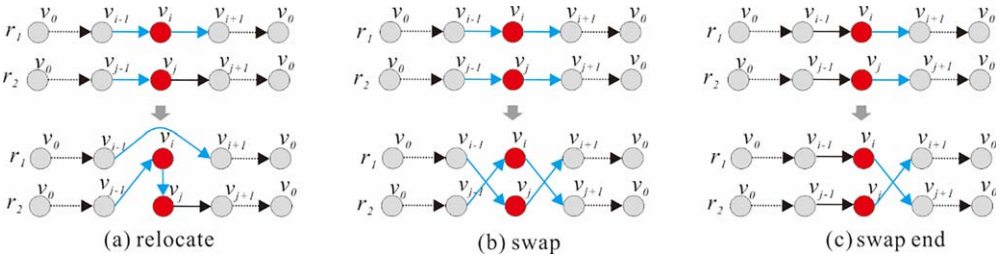

FIGURE 1 Neighborhood structure

some perturbing operations to escape from local minima, including tabu search (Glover, 1977; Jin et al., 2012), simulated annealing (Kirkpatrick, Gelatt, & Vecchi, 1983; Wang et al., 2015), variable neighborhood search (Mladenovic & Hansen, 1997; Polacek et al., 2008), and iterated local search (Lourenço, Martin, & Stützle, 2003; Penna, Subramanian, & Ochi, 2013; Palhazi Cuervo, Goos, Sörensen, & Arráiz, 2014; Silva, Subramanian, & Ochi, 2015).

3.2 | Spatial partitioning of the VRP

Related definitions regarding spatial partitioning of the VRP are given as follows.

Definition 1. The *VRP region R* is a minimum bounding rectangle (MBR) covering all VRP nodes *V*, which can be defined as in Equation 1, where x_{min} and y_{min} are the minimum *x* and *y* coordinate, x_{max} and y_{max} are the maximum *x*, *y* coordinate, respectively.

$$R=\{(x_{\min}, y_{\min}, x_{\max}, y_{\max})|\forall v_i \in V, x_{\min} \le x_{v_i} \le x_{\max}, y_{\min} \le y_{v_i} \le y_{\max}\} \quad (1)$$

Definition 2. Given a VRP instance within the region *R*, a *spatial partitioning* is a division of *R* into a set of spatial cells *A* that covers the whole region without any overlapping or holes.

$$\{R, V\}P \rightarrow \{A|\mathrm{Ua_j}=\mathrm{A},\ \forall \mathrm{i} \neq \mathrm{j}\ \mathrm{a_i Ua_j}=\varnothing\} \quad (2)$$

Definition 3. The *borderline* is a polyline shared by two spatially adjacent cells. Formally, borderline l_{ij} is a polyline shared by spatial cell a_i and a_j.

Definition 4. The *border area* of a spatial cell is the space within a certain distance of the borderline of the cell, which can be defined as in Equation 3, where *p* is a point, b_{ij} denotes the border area around the line l_{ij}, β denotes the distance threshold, and *dist* (*p*, l_{ij}) denotes the distance between *p* and l_{ij}.

$$B_{ij}=\{\mathrm{p}|\mathrm{dist}(\mathrm{p},\ l_{ij})\le\beta,\ \mathrm{p}\in\mathrm{R}\} \quad (3)$$

Taking a rectangular partitioning as an example, Figure 2 illustrates the above definitions. The VRP region (larger than the defined region for clear display) is divided into four spatial cells (a_1 to a_4). Border lines (l_{12}, l_{23}, l_{34}) between spatial cells are represented by dotted lines. Border areas (b_{12}, b_{23}, b_{34}) are the places with a light blue background. Using such useful spatial partitioning, the VRP point set will be divided into many independent subsets within each individual spatial cell, which can be concurrently optimized by local search with multiple computing cores.

Several spatial structures have been developed to group points, such as rectangle, grid, fan, KD-tree, and cluster, which provide us with different spatial divisions. However, different partitioning methods may have different effects on the spatial parallel optimization approach. To investigate the performance of different spatial partitioning strategies, following the above definitions, this article designs five spatial strategies to examine spatial parallel optimization for very large-scale VRP instances, including rectangle, grid, fan-shaped, *k*-dimension (KD) tree, and spatial cluster partitioning.

Definition 5. A *rectangle partitioning* divides the VRP region *R* into a set of rectangles with horizontal/vertical lines. Without loss of generality, the division is performed by a series of horizontal lines y_k as in Equation 4, where *k* is the

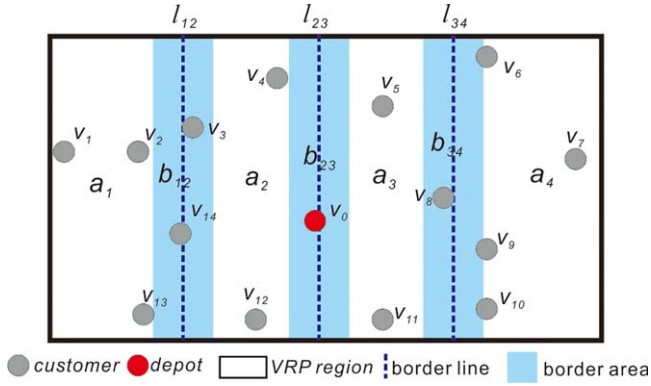

FIGURE 2 Spatial partitioning of a VRP instance

line index, and N is the number of rectangles. Figure 3a illustrates an example with four vertical rectangles. Figure 3b gives another example with four horizontal rectangles.

$$y_k = y_{\min}\left(1-\frac{k}{N}\right)+y_{\max}*\frac{k}{N},\ \ x_k \in [x_{\min}, x_{\max}]\ \ (0<k<N) \tag{4}$$

Definition 6: A *grid partitioning* divides region R into many grid cells with a set of vertical and horizontal lines as in Equation 5, where x_k and y_k denote the vertical and horizontal lines, respectively. As Figure 3c shows, region R is divided into nine grids.

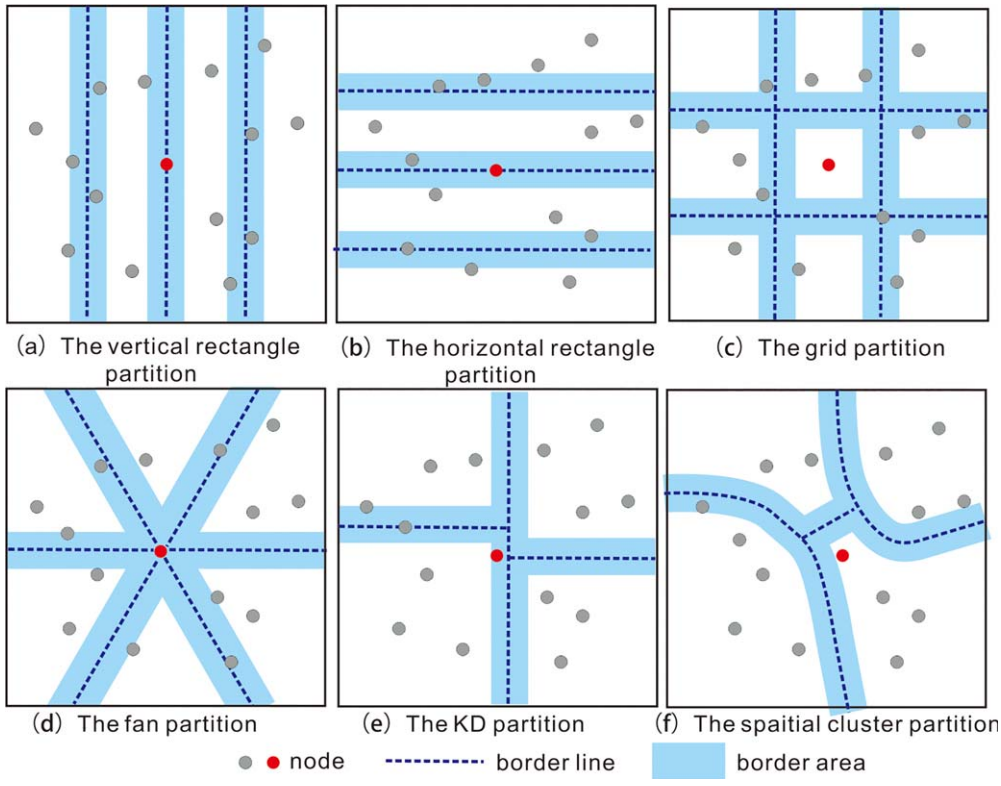

FIGURE 3 Spatial partitioning for the VRP

$$\begin{cases} x_k = x_{\min}\left(1-\frac{k}{N}\right)+x_{\max}\frac{k}{N}, & y_k \in [y_{\min}, y_{\max}] \\ y_k = y_{\min}\left(1-\frac{k}{N}\right)+y_{\max}\frac{k}{N}, & x_k \in [x_{\min}, x_{\max}] \end{cases} \quad \left(0 < k < \sqrt{N}\right) \tag{5}$$

Definition 7: A *fan partitioning* splits region R into fan-shaped cells with many rays starting from the depot. Each ray is defined as in Equation 6, where r is the maximum distance between the customers and the depot. Two sequential rays and the arc between them form a fan-shaped area. Figure 3d illustrates an example of a six-fan division with rays every 60°.

$$\begin{cases} x_k = x_{v_0} + r\cos\left(\frac{2\pi k}{N}\right) \\ y_k = y_{v_0} + r\sin\left(\frac{2\pi k}{N}\right) \end{cases} \quad (r > 0,\ 0 \le k < N) \tag{6}$$

Definition 8: the *k-dimension* (KD) *tree partitioning* (Bentley, 1975) recursively divides a region into two sub-regions with an equal number of nodes using lines, as in Equation 7, where v is the central node in the region, k is the division level, and x^{2k-1} and y^{2k} denote the partitioning lines at the corresponding level; $x_{\min}^{2k-2}$ and $x_{\max}^{2k-2}$ are the minimum and maximum value of the x dimension in the corresponding sub regions, $y_{\min}^{2k-1}$ and $y_{\max}^{2k-1}$ are the minimum and maximum value of the y dimension in the corresponding subregions. As Figure 3e shows, region R is divided into four different sized rectangles that have the same number of nodes by a two-level KD partitioning.

$$\begin{cases} x^{2k-1} = \left(x_{\min}^{2k-2} + x_{\max}^{2k-2}\right)/2, & y \in [y_{\min}, y_{\max}] \\ y^{2k} = \left(y_{\min}^{2k-1} + y_{\max}^{2k-1}\right)/2, & x \in [x_{\min}, x_{\max}] \end{cases} \quad (0 < k < \log_2 N) \tag{7}$$

Definition 9: A *spatial cluster partitioning* divides the region into many arbitrarily shaped spatial cells by grouping nodes into a given number of clusters based on the distance measure. Figure 3f gives an example of four clusters with 14 nodes. This article uses the *k*-means cluster method (Aldstadt, 2008) to group nodes.

3.3 | Spatial parallel optimization for the VRP

The principle of the spatial parallel metaheuristic presented in this article is to use parallel computing to accelerate the VRP solution procedure with the help of spatial partitioning and spatial neighborhood reduction strategies. In the traditional local search-based heuristics in Section 3.1, spatial data dependence does exist because local route changes will lead to side effects on local search evaluation of other nodes. To untangle such interdependent relationships, in cooperation with a spatial neighborhood reduction strategy limiting local search to within a small area (Li et al., 2005; Fang et al., 2013), spatial partitioning is used to divide the VRP region into many spatial cells such that each individual searching process will be limited within a cell without any operation on nodes from other cells. However, because of the borderlines of spatial cells, some promising operating nodes in the border area will be directly rejected during local search, which will decrease the optimization level in this special area. To overcome this negative effect, following the search inside a spatial cell, an alternative local search phase around the borderline is conducted to rearrange the associated route segments to enhance the solution at these places.

Taking the vertical rectangle partitioning as an example, Figure 4 illustrates the principle of the spatial parallel optimization. The VRP region is divided into four spatial cells (a_1 to a_4), and four computing cores/threads (c_1 to c_4) are used to perform concurrent local searches in each spatial cell (Figure 4a) and the border area (Figure 4b) to speed up the local search procedure. Details of the presented spatial parallel optimization framework will be introduced in the next section.

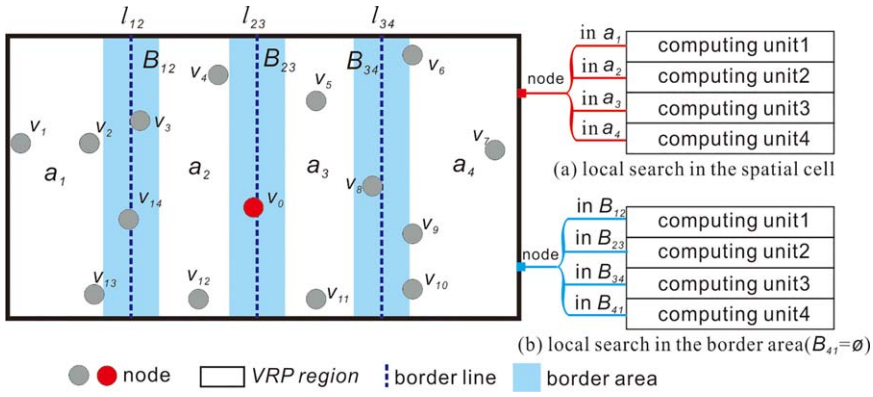

FIGURE 4 Principle of the spatial parallel optimization

4 | DESCRIPTION OF THE SPATIAL PARALLEL OPTIMIZATION ALGORITHM

This section introduces details of the presented novel spatial parallel optimization algorithm. The workflow of the proposed approach is summarized in Figure 5. Given a spatial partitioning strategy, the region of a VRP instance will first be divided into many spatial cells C with a given number, N, which is also equal to the number of computing cores used. Nodes in the border area of each cell will also be identified. Then, an initial solution is created by the parallel insertion of both the spatial cells and border areas. Next, spatial parallel iterated local search (SPILS) is alternatively conducted on nodes in the spatial cells and border areas to improve the solution quality. The iterated local search (ILS) (Lourenço, Martin, & Stützle, 2010), in which the perturbation will be complete when the local search drops into the local minimum, is used to control the improvement procedure. The improvement process will be continued until a number of iterations I_{max} has been reached. It should be noted that both the construction and the improvement are implemented in a parallel computing environment with the help of spatial partitioning. The local search operation will be accelerated with the kth-ring Voronoi neighbors (Fang et al., 2013).

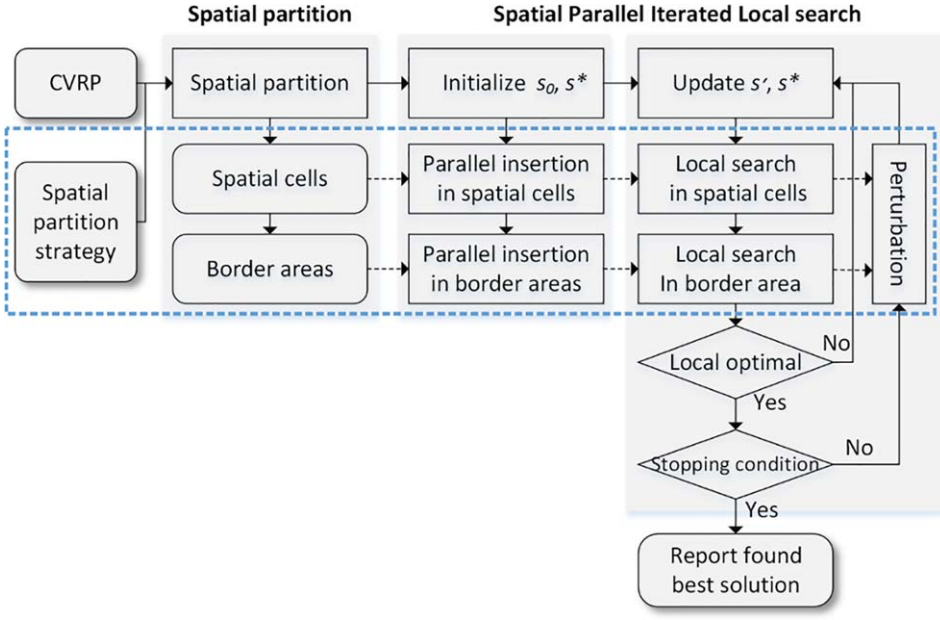

FIGURE 5 Workflow of the spatial parallel iterated local search metaheuristic

Algorithm 1. Iterated Local Search

1 s_0←constructsolution()
2 s^*←localsearch(s_0)
3 **while** *stopping condition is not true*
4 s' ←perturb(s^*)
5 $s^{*'}$ ←localsearch(s')
6 s^* ←acceptancecriterion($s^{*'}$, s^*)
7 **end while**
8 return best found solution s^*

4.1 | Construction algorithm

The initial solution is created using a simple and fast construction algorithm that parallel-inserts unrouted nodes into routes. Each computing unit conducts the node insertion inside a spatial cell a_i. First, a set of empty routes is created. Each route is filled with a seed node that is randomly selected in cell a_i. Then, following a random sequence, the unrouted nodes in cell a_i are inserted into the best places to shorten the total route length. The insertion will be repeated until all nodes in the cell are routed. Next, the nodes in the border area of a spatial cell are inserted. Finally, after the insertion of all nodes, a set of routes visiting all customers represents an initial solution s_0. Because the initial solution s_0 depends on the seed node and the random insertion sequence, to achieve a high-quality initial solution, we run this parallel insertion heuristic N_0 times and keep the best solution.

4.2 | Spatial Parallel Iterated Local Search

The general scheme of SPILS is to parallelize the iterated local search with spatial partitioning. The essential idea of the ILS is to alternately improve the current solution and perturb to escape from local minima (Lourenço et al., 2010). Because of its effectiveness in combinational optimization, it has been widely applied in many VRP variants (Penna et al., 2013; Palhazi Cuervo et al., 2014). However, integration of ILS and spatial parallel computing has not been well investigated. The main steps of ILS are summarized in Algorithm 1. Starting with a high-quality initial solution (*line 1*), the ILS improves the current solution by performing local search (*line 2*). When trapped in a local minimum, instead of generating a new solution to start the improvement again, the solution will be perturbed to escape (*lines 4 and 5*). Only the solution satisfying the acceptance criterion will be preserved (*line 6*). This improvement continues until the stopping condition is true. Finally, the best solution found is reported. *Localsearch* (*lines 2 and 5*) is the main step of ILS. In SPILS, with the help of spatial partitioning, the *localsearch* is concurrently implemented in each spatial cell to achieve spatial parallel optimization.

4.2.1 | Local search

Sequential local search

The implementation of the sequential local search is to make local changes to the current routing with neighborhood structures. The three popular neighborhood structures in Section 3.1 are used in the presented approach. A neighborhood structure *ns* will be randomly selected from the given set (*line 6*). The first chosen node v_i is also randomly generated (*line 7*). Instead of evaluating all the nodes, the second chosen node v_j is limited to be within the kth-ring Voronoi neighbors of node v_i (Fang et al., 2013) (*line 8*). The selected neighborhood structure *ns* evaluates the objective gain (*line 9*). Only a change improving the current solution will be performed (*lines 9 to 11*). For each iteration, this local route change is performed N times (*line 5*), where N denotes the number of customers. The iteration will be repeated until the solution is trapped in a local minimum (*line 12*). The implementation of the local search is summarized in Algorithm 2.

Algorithm 2. Local Search

1 $NS = \{Relocate, Swap, SwapEnd\}$
2 $s' \leftarrow s$
3 **do**
4 $s^* \leftarrow s'$; $s \leftarrow s'$
5 **for** $k=1,\ldots, N//N$ is the number of customer nodes
6 $ns \leftarrow NS$
7 $s' \leftarrow ns(s)$
8 **if** $f(s)<f(s')$
9 $s' \leftarrow s$
10 **end if**
11 **end for**
12 **while** $f(s')<f(s^*)$
13 return s^*

Parallel local search

The use of the kth-ring Voronoi neighbors strategy involves only a few nodes by limiting the second node to be within a small space. Such a local characteristic makes it feasible to conduct spatial parallel searching with spatial partitioning. Instead of evaluating a local route change opportunity at a single place (Algorithm 2, *lines 5 to 11*), using a set of computing units, parallel local search concurrently evaluates route changes at multiple positions to accelerate the improvement.

It should be noted that parallel local search consists of two phases. The first phase focuses on the nodes in a spatial cell a_i (as in Figure 3a). Only the kth-ring Voronoi neighbor of the first chosen node v_i in the same cell will be evaluated. The kth-ring Voronoi neighbors in different cells are directly ignored without evaluation to avoid data dependence in the local search. The number of operations in a iteration will be reduced from N to $N^{'}$, where $N^{'}$ denotes the number of nodes in a spatial cell. The second phase focuses on the nodes in the border area B_{ij} (as in Figure 3b). As opposed to the first phase, the chosen node v_i will be selected from this special area, and the other node is from the kth-ring Voronoi neighbors without any other limitation. The iterations will be reduced from N to $N^{''}$, where $N^{''}$ denotes the number of nodes in the border area of a spatial cell. Because multiple local search operations will be concurrently performed, the computing time will be significantly reduced.

4.2.2 | Perturbation mechanism

The perturbation mechanism of the ILS facilitates escape from a local minimum using larger modifications to a current solution instead of restarting with a new solution. Three types of perturbations are employed in this study, including a perturbation with the minimum route, a perturbation in the border area, and a random perturbation.

- The perturbation with the minimum route destroys the route serving the least nodes in the local best solution so that an unpromising route will be rejected. Nodes in the minimum route will be rejected from the current solution and reinserted back into a proper place in a nearby route. Details of this "destroy-reinsert" mechanism are available in Fang et al. (2013).
- The perturbation in the border area randomly modifies some route segments in the border area around spatial cells. It makes use of *Relocate* to eject a few random nodes in the border area and reinsert them before or after their kth-ring Voronoi neighbors.

- The random perturbation makes use of the neighborhood structure *crossexchange* (Taillard, Badeau, Gendreau, Guertin, & Potvin, 1997) to modify the current local best solution. The crossexchange operator exchanges the positions of two node chains between routes.

The above perturbations are implemented following the roulette-wheel rule. Initially, the three perturbations have equal probability. A random value is generated to determine which perturbation will be performed. During the improvement, the probability will be updated according to their contributions to the objectives as shown in Equation 7, where r_i denotes the probability of the corresponding perturbation, Δs_{pm} denotes the objective improvement of the perturbation with the minimum route, Δs_{pb} denotes the objective improvement of the perturbation in the border area, and Δs_{pr} denotes the objective improvement of the random perturbation.

$$r_i = \frac{\Delta s_i}{\sum_i \Delta s_i} \quad i \in \{pm,\ pb,\ pr\} \tag{7}$$

The number of chosen nodes in the perturbation is the parameter that determines the size of the perturbation, which is the key factor because it defines the degree of the jump from the local minimum. If the perturbation is too small, the following local search tends to fall into the same local minimum. On the other hand, if the perturbation is too large, the better local route structure in the best local solution found will be destroyed, and the following local search has to provide an improved and almost new solution. The calibration of the perturbation size will be discussed in Section 6.2.

5 | EXPERIMENT AND RESULTS

This section introduces the tested datasets, parameter tuning, and the comparison between the SPILS results and the results of the state-of-the-art heuristics and the best known solution (BKS) in the literature.

For easy implementation and low expense, the presented SPILS metaheuristic is coded in C++ and utilizes the shared memory library OpenMP as the parallel computing platform. Using the spatial partitioning strategies (vertical rectangle, horizontal rectangle, grid, fan, KD tree, and cluster) in Section 3.2, six SPILS metaheuristics versions (named VR, HR, GRID, FAN, KD, and Cluster, respectively) were implemented. It should be noted that k-means may generate different clustering results with different initial seeds. We random implement the k-means 20 times and preserve the best clustering result for later spatial parallel optimization. All of them run on a personal computer with 32 computing cores (Intel 2.00 GHz) and 128 GB memory, operating under a Windows 7 64-bit system. The standard configuration of the parallel computing environment (called SPILS standard) makes use of four cores to solve the benchmark VRP instance, which specifies that the VRP region is divided into four cells.

5.1 | Test datasets

Computational experiments were conducted on the one man-made dataset and three real-world large-scale VRP datasets of Zachariadis and Kiranoudis (2010), Fang et al. (2013), and the new super-large-scale VRP in Beijing, China. The man-made dataset contains 10 typical CVRP instances with 30-77 customers from the VRP dataset of Augerat, Belenguer, Benavent, Corberán, Naddef, and Rinaldi (1995) to verify the performance of SPILS. The dataset of Zachariadis and Kiranoudis (2010) has four real-world VRP instances with 3,000 customers to simulate daily logistics services in four major Greece cities. The dataset of Fang et al. (2013) includes four instances in Guangzhou, China with 4,594-8,683 customers and a route length restriction. The new super-large-scale VRP dataset in Beijing, China is derived from very large-scale multi-depot VRP instances (Tu et al., 2014a) by replacing multiple depots with a single depot. Therefore, there are seven super-large VRP instances with 2,000 to 20,000 customers to simulate urban delivery services. For each VRP instance, six SPILS metaheuristics were executed 10 times for the randomness behind the ILS. The

TABLE 1 Parameter settings of the spatial parallel iterated local search metaheuristic

Parameters	Meaning	Candidate values	Value
I_{max}	maximum number of ILS iterations	[10, 100]	20
$I_{perturb}$	size of the perturbation	[0.005N, 0.01N, 0.02N, 0.05N]	0.01N

N denotes the number of nodes in a CVRP instance

total route length and computation time were recorded. The results are compared with exact algorithm and metaheuristics.

5.2 | Parameter tuning

There are two parameters in the presented SPILS metaheuristic listed in Table 1. Parameter I_{max} indicates the time to reach the local minimum, which controls the exit condition of the ILS, while another parameter, $I_{perturb}$, specifies the degree of perturbation. Following the ILS parameter tuning approach of Silva et al. (2015), we calibrated the two parameters with the VRP instances from Beijing, China. First, $I_{perturb}$ is calibrated with a sequential setting of I_{max} within the range [10, 100]. The value that resulted in the best average objective appears at $0.01N$, where N denotes the number of customers. Therefore, we set $I_{perturb}=0.01N$. Then, different stopping conditions I_{max} were tested. A proper value, $I_{max}=20$, was chosen to stably converge to a high-quality solution. We used identical values for solving all instances, as indicated in Table 1. The impact of different parameter settings will be discussed in Section 6.

5.3 | Results for benchmarks of Augerat et al. (1995)

Table 2 compares the results of 10 small VRP instances of Augerat et al. (1995). The first column **Inst** is the instance name. The second column **N** denotes the number of customers, which indicates the size of the VRP. The third column ***S**** denotes the optimal values reported by the branch-and cut algorithm (Baldacci, Christofides, & Mingozzi, 2008). The

TABLE 2 Comparison of results for benchmarks of Augerat et al. (1995)

			SPILS standard (4 cores)					
Inst	N	*S**	VR	HR	GRID	FAN	KD	Cluster
B-n31-k5	30	672[a]	672	672	672	672	672	672
B-n38-k6	35	805[a]	805	805	805	805	805	805
B-n41-k6	40	829[a]	829	829	829	829	829	829
B-n45-k6	44	678[a]	682	682	682	682	682	682
B-n50-k7	49	741[a]	741	741	741	741	741	741
B-n52-k7	51	747[a]	751	751	751	747	747	747
B-n57-k7	56	1,153[a]	1,153	1,153	1,153	1,153	1,153	1,153
B-n63-k10	62	1,503[a]	1,503	1,503	1,503	1,503	1,496	1,503
E-n67-k10	66	1,036[a]	1,036	1,032	1,036	1,036	1,036	1,036
E-n78-k10	77	1,237[a]	1,237	1,237	1,221	1,221	1,221	1,237
Mean Dev to *S** **(%)**			**0.33**	**0.37**	**0.33**	**0.20**	**0.16**	**0.09**
AVG time (/minutes)			**0.19**	**0.19**	**0.19**	**0.19**	**0.2**	**0.23**

The standard parallel computing environment is with four computing cores. S*: optimal solution. [a]Baldacci et al. (2008)

TABLE 3 Comparison of results for VRP dataset in Greece

					SPILS standard					
Inst	**N**	**BKS**	**PSMDA**	**VSHA**	**HR**	**VR**	**GRID**	**FAN**	**KD**	**Cluster**
Zk1	3,000	13,532.12[a]	13,666.36	13,532.12	13,603.00	13,599.90	13,614.90	13,613.70	13,600.30	13,589.20
Zk2	3,000	3,482.56[a]	3,536.25	3,482.56	3,486.61	3,487.79	3,494.02	3,496.41	3,487.50	3,487.70
Zk3	3,000	1,145.28[a]	1,170.33	1,145.28	1,163.81	1,162.35	1,162.67	1,162.45	1,158.55	1,159.43
Zk4	3,000	1,119.90[a]	1,139.08	1,119.90	1,131.66	1,130.71	1,133.09	1,131.17	1,129.34	1,127.06
Mean Dev to BKS (%)			**1.61**	**0.00**	**0.83**	**0.78**	**0.91**	**0.88**	**0.66**	**0.61**
AVG time (/minutes)			**172.2**	**23.4**	**2.1**	**2.3**	**2.1**	**2.0**	**2.0**	**2.3**

*The standard parallel computing platform is with 4 computing processors. [a]Fang et al. (2013)

remaining six columns denote the results of the SPILS framework using six partitioning strategies. Each entry reports the average total route length. The average computing time is summarized in the last row.

The results indicate that SPILS achieves high quality solutions in these small VRP instances. The gaps between SPILS standard and optimal solutions are no more than 0.37%. SPILS Cluster holds the minimum gap (0.09%) while SPILS HR has the maximum gap (0.37%). It demonstrates that the SPILS has a good ability to find the high quality solution. Regarding the computing time, the SPILS standard averagely consumes no more than 0.24 minutes for VRP instances with less than 78 customers.

5.4 | Results for real-world large-scale VRP of Greece

Table 3 compares the results of VRP instances in Greece with two state-of-art metaheuristics, the PSMDA developed by Zachariadis and Kiranoudis (2010) and the VSHA presented by Fang et al. (2013). The third column **BKS** denotes the best known solution reported by state-of-the-art metaheuristics. The fourth and fifth columns provide the results reported by PSMDA and VSHA, respectively. It shows that SPILS outperforms the PSMDA in both solution quality and computing efficiency. In terms of the deviation from the BKS, the SPILS standard (0.61-0.91%) is superior to that of Zachariadis and Kiranoudis (2010) (1.61%) but inferior to the VSHA (0.0%). Regarding the computing time, the SPILS standard consumes only 9.8% (=2.3/23.4) of the time used by VSHA. Taking ZK1 as an example, Figure 6 displays the solution that was obtained. It suggests the effectiveness of cluster-based spatial partition as near customers are visited by the same vehicle.

5.5 | Results for real-world very large-scale VRP in Guangzhou, China

Table 4 describes the results for four real-world very large-scale VRP instances in Guangzhou, China. The results reported by VSHA are also shown for comparison. The format is similar to that of Table 2. For these super-large-scale VRP instances with 4,594-8,683 customers, the deviation from the BKS of the solution reported by SPILS decreases to the range of [0.33%, 0.66%]. This indicates that the difference between the SPILS standard and the VSHA is significantly narrowed. In terms of computing time, the SPILS costs no more than 9.8% (=5.2/53.0) of the computing effort of the VSHA.

5.6 | Results for real-world very large-scale VRP in Beijing, China

Table 5 reports the results of VSHA and the SPILS standard for super-large-scale VRP instances in Beijing, China. It is noticeable that the SPILS standard achieves similar quality solutions to those of VSHA for such super-large-scale VRP instances with much less computing effort spent. The maximum gap between the SPILS standard and the VSHA is reduced to 0.19% for the vertical rectangle and horizontal rectangle strategies. The minimum gap is only −0.02%, as

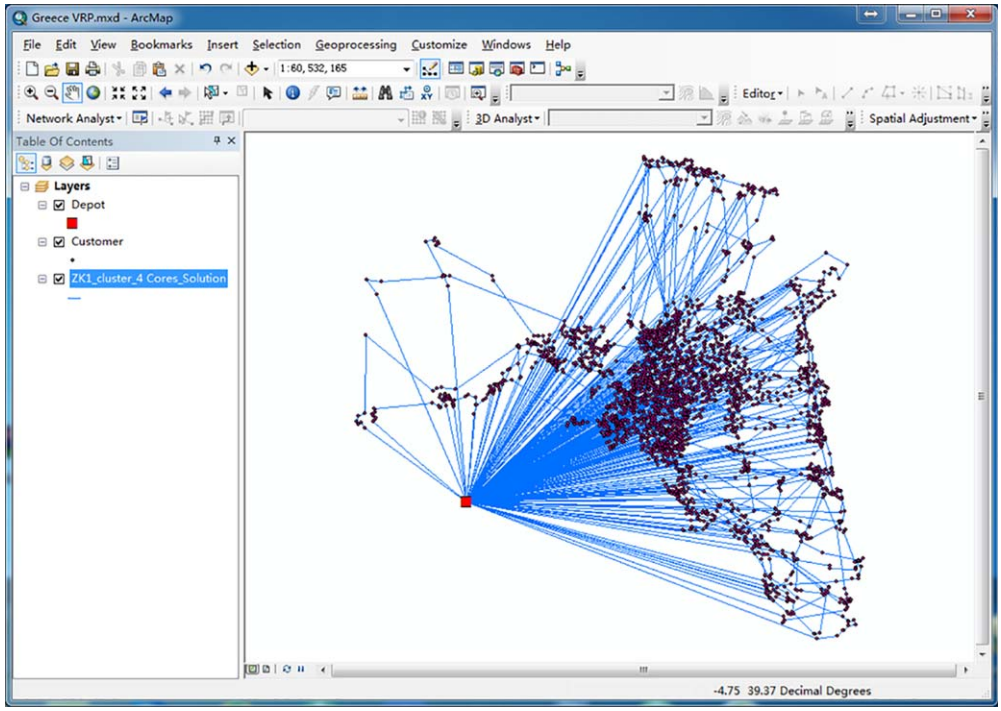

FIGURE 6 Solution of ZK1 with SPILS cluster in Greece

reported for the k-dimension tree (KD) and the cluster strategy. With respect to computing efficiency, SPILS consumes 9.6% (=23.0/243.8) of the time used for VSHA.

5.7 | Comparison between six spatial partitioning strategies

Table 6 summarizes the performance of the SPILS framework on different sized VRPs. Each entry reports the average deviation (%) from the BKS. It demonstrates that all SPILS versions exhibit better performance as the size of CVRP becomes larger and larger. For the VRP dataset in Greece with 3,000 customers, the minimum gap from the BKS is 0.61%, reported by the clustering strategy. For the largest Beijing dataset with 2,000-20,000 customers, the minimum gap from the BKS is reduced to −0.02%, which is generated by the KD tree and clustering strategies.

It can be observed that the SPILS with the clustering strategy achieves the best performance. On the Greece, Guangzhou, and Beijing datasets, the average deviations from the BKS reported by the SPILS with the clustering strategy are 0.61, 0.33 and −0.02%, respectively. The reason may be the fact that clustering keeps a compact spatial

TABLE 4 Comparison of results for VRP dataset in Guangzhou, China

				SPILS standard					
Inst	N	BKS	VSHA	VR	HR	GRID	FAN	KD	Cluster
GZ1	4,594	3,362,094.6 [a]	3,362,094.6	3,385,422.7	3,393,339.1	3,380,102.9	3,380,102.9	3,373,346.3	3,367,669.8
GZ2	4,685	3,244,528.5 [a]	3,244,528.5	3,271,965.5	3,253,858.8	3,269,302.6	3,269,302.6	3,257,315.8	3,262,088.7
GZ3	8,222	5,920,408.6 [a]	5,920,408.6	5,951,127.8	5,940,742.5	5,972,157.0	5,972,157.0	5,953,090.5	5,937,497.5
GZ4	8,683	6,083,290.9 [a]	6,083,290.9	6,109,531.7	6,130,782.3	6,112,239.4	6,112,239.4	6,094,176.8	6,104,228.8
Mean Dev to BKS			**0.0%**	**0.62**	**0.59**	**0.66**	**0.66**	**0.36**	**0.33**
AVG time (/minutes)			**53.0**	**5.0**	**5.1**	**5.2**	**5.0**	**4.9**	**4.9**

*The standard parallel computing platform is with 4 computing cores. [a]Fang et al. (2013)

TABLE 5 Results for super large VRP dataset in Beijing, China

			SPILS standard (Four cores)					
Inst	**N**	**VSHA (/km)**	**VR**	**HR**	**GRID**	**FAN**	**KD**	**Cluster**
BJ1	2000	4,000.1	4,029.5	4,013.7	4,027.7	4,004.6	4,016.1	4,003.2
BJ2	3,000	5,242.3	5,290.4	5,260.7	5,255.1	5,276.7	5,245.7	5,255.0
BJ3	5,000	8,024.9	**8024.0**	8,056.4	**8,006.3**	**7,978.3**	**7,995.8**	**7,990.7**
BJ4	8,000	12,083.1	12,136.3	12,086.7	12,089.5	12,085.0	**12,062.1**	12,098.5
BJ5	12,000	17,199.1	17,236.1	17,267.7	17,225.8	17,204.5	17,229.4	17,210.2
BJ6	16,000	22,253.1	**22,217.7**	**22,204.0**	**22,215.3**	**22,191.2**	**22,216.8**	**22,191.8**
BJ7	20,000	27,113.2	**27,089.2**	27,120.1	27,120.4	**27,113.7**	**27,086.6**	27,120.5
Mean Dev to VSHA(%)		**0.0**	**0.19**	**0.19**	**0.11**	**−0.01**	**−0.02**	**−0.02**
Mean time (/minutes)		**243.8**	**23.3**	**22.7**	**23.0**	**23.0**	**22.1**	**22.7**

*The standard parallel computing platform is with four computing cores. **Bold** indicates a better solution

structure within a cluster but a loose connection between clusters, which mirrors the principle of parallel computing. It suggests that geographical knowledge regarding customer distribution can enhance the optimization performance of local search-based heuristics. It also indicates that spatial partitioning with the vertical rectangle (VR), horizontal rectangle (HR), grid (GRID), and the fan (FAN) achieves similar quality solutions, as their average deviations are above 0.5%.

6 | DISCUSSION

This section investigates the speedup performance, the impact of the perturbation, and the trade-off between solution quality and computing effort.

6.1 | Speedup performance

The speedup performance is one of the most widely used metrics to measure the effectiveness of a parallel metaheuristic, and it is defined as the ratio between the efficiencies of the sequential and parallel algorithms (Ananth, George, Vipin, & Anshul, 2003). To examine this metric, we developed a sequential version of the presented SPILS metaheuristic by removing the spatial parallel mechanism from the local search in Figure 5. Experiments were carried out on seven real-world large-scale instances in Beijing, China derived from Tu, Li, and Fang (2014b), with customers varying from 2,000 to 20,000. The number of used computing processors varies from two to 32, which specifies that the whole VRP region must be divided into the same number of spatial cells. The speedup value for the GRID does not exist at two, eight, and 32 because they are not roots of an integer. The results are shown in Table 7. Each entry denotes the

TABLE 6 Average deviation to the best known result of the SPILS

		SPILS standard					
Test datasets	**N**	**VR**	**HR**	**GRID**	**FAN**	**KD**	**Cluster**
Greece Dataset (%)	3,000	0.83	0.78	0.91	0.88	0.66	0.61
Guangzhou Dataset (%)	4,594-8,683	0.62	0.59	0.66	0.66	0.36	0.33
Beijing dataset (%)	2,000-20,000	0.19	0.19	0.11	**−0.01**	**−0.02**	**−0.02**
Mean (%)		**0.55**	**0.52**	**0.56**	**0.51**	**0.33**	**0.31**

TABLE 7 Speedup performance of the SPILS

	SPILS					
Num of cores	VR	HR	GRID	FAN	KD	Cluster
2	1.83	1.77	-	1.81	1.92	1.74
4	3.32	3.43	3.46	3.59	3.76	3.42
8	5.05	5.13	-	5.32	5.58	5.43
16	7.41	7.59	7.74	8.26	8.53	7.52
32	11.01	11.17	-	12.39	13.00	12.11

speedup value of a spatial partitioning strategy with the given computing cores. It indicates that the SPILS exhibits good scalability for each spatial partitioning strategy. With the increase of computing cores, the speedup performance becomes better and better. It can be observed that the KD tree partitioning has the best speedup performance because of the equal number of cells in this recursive partitioning.

6.2 | Impact of the perturbation

To examine the performance of the perturbation, taking the largest Beijing CVRP dataset as an example, the impact of the size of the perturbation was tested with different settings of the size perturbation. Figure 7 displays the deviation from the BKS. We can observe that the perturbation setting of 0.01N achieves the best solutions. A smaller perturbation (0.005N) results in worse solutions (0.12%), as it could lead to a jump back to the same local minimum, while a larger perturbation (0.05N) pushes the perturbed solution far away from the current solution and generates the worst results (0.23%). It also verifies the correctness of the setting of the perturbation size in Section 5.2.

6.3 | Trade-off between solution quality and computing effort

To further evaluate the trade-off between solution quality and computing effort, taking the instance BJ4 as an example, the SPILS standard was executed with different stopping conditions as the iteration number varied from 10 to 18. The SPILS standard was run 10 times. The average deviation of the SPILS average results from the BKS was calculated and is shown in Table 8. The computing time per spatial partitioning strategy was also averaged; the results are shown in Table 8. This demonstrates that the SPILS solution gradually improves as the computing effort increases until a peak level is reached. It also indicates that it is proper to set parameter I_{max} to 20 because the results have converged at iteration 16.

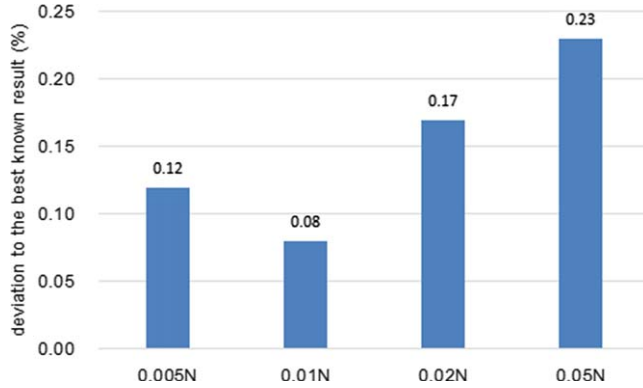

FIGURE 7 Impact of the perturbation size

TABLE 8 Comparison of the trade-off between solution quality and computing effort

l_{max}	10	12	14	16	18
Avg deviation from average result to BKS (%)	0.25	0.12	0.05	0.00	0.00
Avg computing time **(/minutes)**	13.6	17.2	19.6	20.6	22.1

7 | CONCLUSIONS

Reducing the computing time for large-scale VRPs is a major challenge in many real-world transportation applications. From a spatial view, this article presents easy-to-implement, highly effective spatial parallel iterated local search to address the very large-scale vehicle routing problem. As opposed to traditional local search methodologies focusing on the solution space, the presented approach parallelizes the local route changes with the help of spatial partitioning and spatial neighborhood reduction strategies. Spatial partitioning is devised to decompose the VRP region into a set of small spatial cells, and therefore it facilitates concurrent local search to accelerate the solution procedure. Additional local search in the border around spatial cells is used to handle associated spatial data dependence by spatial parallel local search. Iterated local search is used to embed this efficient spatial parallel optimization process and escape from local minima. Six spatial partitioning strategies, consisting of vertical rectangle, horizontal rectangle, grid, fan, KD tree, and cluster, were tested.

Computational experiments on large-scale VRP datasets with 2,000 to 20,000 customers demonstrate good performance of the presented spatial parallel heuristics approach. This method is also very flexible in the sense that it significantly reduces the computing time on different-sized VRP instances without much loss of solution quality. It indicates that the cluster strategy usually generates the best solution, while the KD tree partitioning has the best efficiency. It implies that geographical knowledge supports the optimization performance of local search-based heuristics.

The contribution of this research can be summarized as being twofold. First, a novel spatial parallel optimization approach integrating spatial partitioning, local search, and parallel computing is proposed for the solving of very large-scale VRPs. It sheds light on the hybridization of modern optimization methodology, spatial thinking, and parallel computing. Second, very large-scale VRPs with up to thousands of customers are efficiently and effectively solved in less than half an hour. For less big VRPs, the effort of SPILS and VSHA is comparable. But SPILS consumes much less computing time. Such advances are beneficial to transportation cases with ever-growing sizes in real-world applications.

In this article, we focused on the spatial parallelization of state-of-the-art local search-based metaheuristics for large-scale VRPs. Further work can be conducted on many aspects. First, we could focus on the cooperation between the spatial decomposition and solution decomposition mechanisms. Second, the presented effective SPILS can be integrated with cyber infrastructure and cloud GIS to provide fast and effective geographical information services in the field of transportation (Li et al., 2016).

ACKNOWLEDGMENTS

This research was supported in part by the National Science Foundation of China (#41401444, #41501424, #91546106), Natural Science Foundation of SZU (#2016065), Natural Science Foundation of Guangdong Province (#2016A030310168), Open Research Fund Program of Shenzhen Key Laboratory of Spatial Smart Sensing and Services (Shenzhen University) (#42990005), and Shenzhen Basic Research Program (JCYJ20140828163633980).

REFERENCES

Aldstadt, J. (2008). Spatial clustering. In M. M. Fischer & A. Getis (Eds.), *Handbook of Applied Spatial Analysis*. (pp. 279–300). Berlin, Germany: Springer.

Ananth, G., George, K., Vipin, K., & Anshul, G. (2003). *Introduction to parallel computing*. Harlow, UK: Pearson.

Augerat, P., Belenguer, J, M., Benavent, E., Corberán, A., Naddef, D., & Rinaldi, G. (1995). *Computational results with a branch and cut code for the capacitated vehicle routing problem*. Grenoble, France: Université Joseph Fourier Research Report No. 949-M.

Arribas, C. A., Blazquez, C. A., & Lamas, A. (2010). Urban solid waste collection system using mathematical modelling and tools of geographic information systems. *Waste Management Research*, *28*, 355–363.

Artés, T., Cencerrado, A., Cortés, A., & Margalef, T. (2016). Real-time genetic spatial optimization to improve forest fire spread forecasting in high-performance computing environments. *International Journal of Geographical Information Science*, *30*(3), 594–611.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, *18*, 509–517.

Baldacci, R., Christofides, N., & Mingozzi, A. (2008). An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, *115*(2), 351–385.

Curtin, K. M., Voicu, G., Rice, M. T., & Stefanidis, A. (2014). A comparative analysis of travelling salesman solutions from geographic information systems. *Transactions in GIS*, *18*(2), 286–301.

Djidjev, H., Chapuis, G., Andonov, R., Thulasidasan, S., & Lavenier, D. (2015). All-pairs shortest path algorithms for planar graph for GPU-accelerated clusters. *Journal of Parallel & Distributed Computing*, *85*, 91–103.

Fang, Z. X., Tu, W., Li, Q. Q., Shaw, S. L., Chen, S., & Chen, B. Y. (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.

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Sciences*, *8*(1), 156–166.

Golden, B., Raghavan, S., Wasil, E. A. (2008). *The vehicle routing problem latest advance and new challenges*. Berlin, Germany: Springer.

Groër, C., Golden, B., & Wasil, E. (2011). A parallel algorithm for the vehicle routing problem. *INFORMS Journal of Computing*, *23*, 315–330.

Janssens, J., Van den Berghm, J., Sörensen, K., & Cattrysse, D. (2015). Multi-objective microzone-based vehicle routing for courier companies: From tactical to operational planning. *European Journal of Operation Research*, *242*, 222–231.

Jin, J. Y., Crainic, T. G., & Lokketangen, A. (2012). A parallel multi-neighborhood cooperative tabu search for capacitated vehicle routing problems. *European Journal of Operational Research*, *222*, 441–451.

Jin, J. Y., Crainic, T. G., & Løkketangen, A. (2014). A cooperative parallel metaheuristic for the capacitated vehicle routing problem. *Computers & Operations Research*, *44*, 33–41.

Jung, H., Lee, K., & Chun, W. (2006). Integration of GIS, GPS, and optimization technologies for the effective control of parcel delivery service. *Computers & Industrial Engineering*, *51*, 154–162.

Kinobe, J. R., Bosona, T., Gebresenbet, G., Niwagaba, C. B., & Vinneras, B. (2015). Optimization of waste collection and disposal in Kampala city. *Habitat International*, *49*, 126–137.

Kirkpatrick, S., Gelatt, Jr. C. D., & Vecchi, M. P. (1983) Optimization by simulated annealing. *Science*, *220*, 671–680.

Kuo, P. F., Lord, D., & Walden, T. D. (2013). Using geographical information systems to organize police patrol routes effectively by grouping hotspots of crash and crime data. *Journal of Transport Geography*, *30*, 138–148.

Laporte, G. (2009). Fifty Years of Vehicle Routing. *Transportation Science*, *43*, 408–416.

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

Li, Q., Chen, B. Y., Wang, Y., & Lam, W. H. K. (2015). A hybrid link-node approach for finding shortest paths in road networks with turn restrictions. *Transactions in GIS*, *19*, 915–929.

Li, W., Cao. K., & Church, R. L. (2016). Cyberinfrastructure, GIS, and spatial optimization: opportunities and challenges. *International Journal of Geographical Information Science*, *30*(3), 427–431.

Lourenço, H. R., Martin, O., Stützle, T. (2003) Iterated Local Search. In F. Glover & G. Kochenberger (Eds.), *Handbook of metaheuristics* (pp. 321–353). New York, NY: Kluwer Academic Publishers International Series in Operations Research & Management Science Vol. 57.

Lourenço, H. R., Martin, O., & Stützle, T. (2010). Iterated local search: Framework and applications. In M. Gendreau & J. Y. Potvin (Eds.), *Handbook of metaheuristics*. (pp. 363–397). Berlin, Germany: Springer.

Mendoza, J. E., Medaglia, A. L., & Velasco, N. (2009). An evolutionary-based decision support system for vehicle routing: The case of a public utility. *Decision Support System*, *46*, 730–742.

Miller, H. J., & Shaw, S. L. (2001). *Geographic information systems for transportation: Principles and applications*. New York, NY: Oxford University Press.

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

Ouyang, Y. (2007). Design of vehicle routing zones for large-scale distribution systems. *Transportation Research Part B: Methodological*, *41*, 1079–1093.

Palhazi Cuervo, D., Goos, P., Sörensen, K., & Arráiz, E. (2014). An iterated local search algorithm for the vehicle routing problem with backhauls. *European Journal of Operational Research*, *237*, 454–464.

Penna, P. H. V., Subramanian, A., & Ochi, L. S. (2013). An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, *19*, 201–232.

Polacek, M., Benkner, S., Doerner, K. F., & Hartl, R. F. (2008). A cooperative and adaptive variable neighborhood search for the multi depot vehicle routing problem with time windows. *BuR-Business Research*, *1*, 207–218.

Qi, M., Lin, W. H., Li, N., & Miao, L. (2012). A spatiotemporal partitioning approach for large-scale vehicle routing problems with time windows. *Transportation Research Part E: Logistics and Transportation Review*, *48*, 248–257

Qin, C. Z., Zhan, L. J., Zhu, A. X., & Zhou, C. H. (2015). A strategy for raster-based geocomputation under different parallel computing platforms. *International Journal of Geographical Information Science*, *28*(11), 2127–2144.

Subramanian, A., Drummond, L. M. A., Bentes, C., Ochi, L. S., & Farias, R. (2010). A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, *37*(11), 1899–1911.

Silva, M. M., Subramanian, A., & Ochi, L. S. (2015). An iterated local search heuristic for the split delivery vehicle routing problem. *Computers & Operations Research*, *53*, 234–249.

Taillard, É., Badeau, P., Gendreau, M., Guertin, F., & Potvin, J. Y. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, *31*, 170–186.

Toth, P., & Vigo, D. (2003). The granular tabu search and its application to the vehicle-routing problem. *INFORMS Journal on Computing*, *15*, 333–346.

Toth, P., & Vigo, D. (2014). *Vehicle routing: Problems, methods, and applications* (2nd Ed.). Philadelphia, PA: Society for Industrial and Applied Mathematics.

Tu, W., Fang, G. Z., Li, Q., Shaw, S. L., & Chen, B. Y. (2014a). A bi-level Voronoi diagram-based metaheuristic for a large-scale multi-depot vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review*, *61*, 84–97.

Tu, W., Li, Q., Chang, X., Yue, Y., & Zhu, J. (2015a). A Spatio-temporal decision support framework for large scale logistics distribution in metropolitan areas. In F. Harvey & Y. Leung (Eds.) *Advances in spatial data handling* (pp. 193–206). Berlin, Germany: Springer.

Tu, W., Li, Q., & Fang G. Z. (2014b). Large scale multi-depot logistics routing optimization based on network Voronoi diagram. *Acta Geodaetica et Cartographica Sinica*, *43*(10), 1075–1082.

Tu, W., Li, Q., Fang, G. Z., & Zhou, B. (2015b). A novel spatial-temporal Voronoi diagram-based heuristic approach for large-scale vehicle routing optimization with time constraints. *ISPRS International Journal of Geo-Information*, *4*(4), 2019–2044.

Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2013). Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. *European Journal of Operational Research*, *231*, 1–21.

Wang, C., Mu, D., Zhao, F., & Sutherland, J. W. (2015). A parallel simulated annealing method for the vehicle routing problem with simultaneous pickup–delivery and time windows. *Computers & Industrial Engineering*, *83*, 111–122.

Weigel, D., & Cao, B. (1999). Applying GIS and OR techniques to solve sears technician-dispatching and home-delivery problems. *Interfaces*, *29*, 112–130.

Xue, W., & Cao, K. (2016). Optimal routing for waste collection: A case study in Singapore. *International Journal of Geographical Information Science*, *30*(3), 554–572.

Zachariadis, E. E., & Kiranoudis, C. T. (2010). A strategy for reducing the computational complexity of local search-based methods for the vehicle routing problem. *Computers & Operations Research*, *37*(12), 2089–2105.

Zhang, J. L., Lam, W. H. K., & Chen, B. Y. (2013). A stochastic vehicle routing problem with travel time uncertainty: Trade-off between cost and customer service. *Networks & Spatial Economics*, *13*(4), 471–496.

Zhang, J., Lam, W. H. K., & Chen, B. Y. (2016). On-time delivery probabilistic models for the vehicle routing problem with stochastic demands and time windows. *European Journal of Operational Research*, *249*(1), 144–154.

How to cite this article: Tu W, Li Q, Li Q, Zhu J, Zhou B, Chen B. A spatial parallel heuristic approach for solving very large-scale vehicle routing problems. *Transactions in GIS*. 2017;21:1130–1147. https://doi.org/10.1111/tgis.12267