Unknown date

Tu, Wei ; Li, Qingquan ; Fang, Zhixiang ; Zhou, Baoding

IJGI

1. Introduction

Flexible transportation service not only reduces travel cost, but also alleviates related energy and environmental concerns, such as traffic congestion, energy consumption, and carbon emission [1,2]. With the advancement of geographic information science (GIS), space-related problem decision-making has been embedded in spatial decision support systems (SDSSs) to provide flexible transportation service in both public and private sectors [3], such as the design of school bus routes [4], collection of solid waste [5,6], distribution of goods in chain business [7], design of police patrol routes [8], and planning express deliveries [9]. Spatial function tools are used for intelligent transportation operation in many services including management of geo-referenced transport network data, geocoding massive human demands, positioning moving vehicles, and navigating routes for drivers [5,6,7,8,9].

Vehicle routing optimization (VRO) aims to design least cost routes to satisfy geographically-distributed human demand through spatial intelligence of SDSSs in the field of transportation [10], including the shortest path problem (SPP) [11,12,13], traveling salesman problem (TSP) [14], and the vehicle routing problem (VRP) [5,6,7]. One type of VRO is modeled as the vehicle routing problem with time window (VRPTW), which imposes time constraints on customers, vehicles, and depots to accept or distribute high quality service [4,6,14]. It specifies that a customer only accepts certain transportation-related service in a given time interval [15,16,17]. The VRPTW is computationally intractable in many real-life applications [18]. The heuristic algorithm is the only justified approach as it can find good solutions in a reasonable time [16,17]. There are two types of heuristic algorithms for the VRPTW, which include local search and evolutionary optimization. Local search iteratively modifies local route structures to improve solution quality and follows a single trajectory to explore the solution space [3,6,7]. Evolutionary optimization simultaneously searches multiple parts of the whole solution space to converge on the best solution [5,19,20]. Recently, state-of-the-art heuristic algorithms have been reported to yield a high quality solution of the VRPTW with hundreds of customers [5,9,18]. More efficient heuristics are needed for even larger VRPTW applications.

Spatial principle has exhibited good performance in space-related problem decision-making [21,22,23] in problems such as bus stop location [3], hazmat route planning [20], and path covering [24]. To efficiently solve VRO, from a spatial perspective, a few speed-up strategies have been developed to handle spatial issues and extend solvable VRO instances up to thousands of customers, which include the granular neighborhood [25], the

kth nearest neighbors [26], and the k-ring-shaped Voronoi neighbors [27]. The common denominator of these effective strategies is to search only a limited number of spatial neighbors to save computing efforts [28]. However, such useful strategies become ineffective for the VRPTW since time constraint imposes an exceptional influence on the proximity of consecutively served customers in a route. A long wait will occur if two spatially-close customers with quite different time windows (for example, one is 8:00–9:00 a.m. and the other is 3:00–4:00 p.m.) are visited consecutively in a route, which challenges the motivation of spatial proximity-based heuristics approaches. Hence, spatial proximity measures must be extended to address the additional temporal constraints [18]. Moreover, how to accelerate the VRO with the help of spatial-temporal thinking should be investigated.

Voronoi diagrams describe both spatial proximity and topological proximity between spatial objects [29,30,31,32]. By extending it to the space-time context, a spatial-temporal Voronoi diagram (STVD) is well suited for measuring the proximity of customers with time constraints. Therefore, this article proposes a spatial-temporal Voronoi diagram-based heuristic approach to solve very large-scale VRPTWs. A derived spatial-temporal Voronoi neighbors from the Voronoi diagram is used to select promising candidates for a local search. Furthermore, motivated by the truth that nearer neighbors have a higher possibility to be consecutively served in a route [28], a Voronoi distance decay strategy is proposed to assign reasonable search efforts on neighbors with different Voronoi distances. A spatial-temporal feature guided search is used to reconstruct unpromising micro route structures. Intensive experiments on benchmark datasets (25–1000 customers) and real-world large scale VRPTWs and its multi-depot variants (2000–10,000 customers) have been implemented to assess the performance of the proposed approach. This efficient and effective approach enables current local search heuristics to solve super large-scale VRPTWs and will contribute to improving spatial intelligence for transportation-oriented SDSSs.

The remainder of this paper is organized as follows. Section 2 reviews related literature. Section 3 introduces the VRPTW, the STVD, the derived spatial-temporal Voronoi distance and neighbors. The proposed spatial-temporal Voronoi diagram-based heuristic for the VRPTW and its multi-depot variants are described in Section 4, and details of the experiments, results, and comparisons are reported in Section 5. Section 6 discusses the impact of the STVD and spatial-temporal proximity behind the best found solutions. The last section draws the conclusions.

2. Literature Review

This section reviews heuristic algorithms for the VRPTW and the integration of vehicle routing optimization and GIS.

2.1. Heuristic Algorithms for the VRPTW

Heuristic algorithms for VRPTWs are divided into two categories: local search and evolutionary optimization. Starting from an initial solution, local search shifts from a current solution to a better neighborhood solution

with a simple change of local routes [33]. The change is made iteratively by moving nodes or exchanging arcs within a route or between routes such as 1–0 exchange, 1–1 exchange, 2-Opt, and λ-opt. Usually, the change process becomes trapped in a local optimum. To overcome unpromising results, many intelligent strategies that accept some worse solutions have been developed to further improve the solution quality, such as simulated annealing [34], iterated local search [35], large neighborhood search [36], variable neighborhood search [37], and tabu search [38]. Numerous tests on small- and medium-size VRPTWs with 25–100 customers have demonstrated good performance of local search heuristics [34,36,37].

Evolutionary optimization concurrently improves many solutions and results in high quality solutions. Four major steps are involved: representation, selection, combination, and mutation. Some of the most successful evolutionary optimization algorithms for the VRPTW are provided by Repoussis et al. [39] and Vidal et al. [40]. Mester and Bräysy [41] used the standard evolutionary optimization framework to guide exploration in the VRPTW solution space with solution initialization and evolution. Gehring and Homberger [42] parallelized the genetic algorithm and first solved VRPTW instances up to 1000 customers. Usually, local search is used as a component in the evolutionary optimization approach. Fast and simplified node or arc exchange-based local search have been used for offspring education in the genetic algorithm [41]. Therefore, more efficient local search heuristics also help to improve evolutionary optimization.

For larger VRPTW instances, recent advances have proposed several accelerating techniques to save computing efforts while still achieving high quality solutions. Following the local search principle, taking spatial distance into account, Cordeau [38] limited the local search candidates with kth nearest neighbors to shorten computing time. Toth and Vigo [25] selected the local search evaluated nodes with a fixed distance threshold. Considering time constraints, Ibaraki [43] proposed a time-oriented neighbor list-based strategy to reduce the local search evaluation. Some unpromising search candidates had been excluded with the unmatched time windows. But the spatial proximity has been ignored. Nagata [44] developed a time warp operation for customers with a late service to avoid complex time window analysis. An additional penalty function was added to the original objectives to justify the local search heuristics. However, the spatial dimension and the temporal dimension are still separated in these local search heuristics.

Another effective accelerating approach is to decompose large-scale VRPTW instances into many smaller subproblems. Using spatial clustering, Dondo and Cerdá [45] divided the large scale VRPTW into a set of small-sized problems. Routes were generated for each small problem to provide a high quality initial solution. Qi et al. [18] developed an effective spatial-temporal distance measure to deal with both space and time issues. A large-scale VRPTW was partitioned with a k-medoid clustering method to generate a good initial solution. A standard genetic algorithm was used to improve the initial solution. Good performance of these effective approaches has been verified by VRPTW instances with no more than 1000 customers.

With regard to the Voronoi diagram, the only work has been provided by Milthers [46] who used two-dimensional Voronoi diagrams to split the VRPTW into many subproblems and then solve them with large neighborhood search heuristics. It showed the effectiveness of Voronoi diagrams in guiding the search process. However, this study dealt only with decomposition of the VRPTW in the solution construction stage. Use of the spatial-temporal Voronoi diagram in speeding up the local search for the VRPTW should still be further considered.

2.2. Integration of Vehicle Routing Optimization and GIS

Effective approaches to find high quality solutions for VRPTWs have been embedded in a GIS-based SDSS to deal with many real world applications. Generally, the integration between GIS and vehicle routing optimization is tightly coupled. Spatial data management, processing, and visualization tools are used to collect customer orders, georeference related data, activate the solving process, and display routes for VROs, as in GIS software such as ArcGIS and TransCAD. In line with local search, Weigel and Cao [7] first reported their efforts in the implementation of a tabu heuristics approach in a GIS environment to deal with VRO for a major American retailer. Following the evolutionary optimization line, Mendoza et al. [5] integrated a customized routing module, which improved solution quality with commercial solutions such as SAP/R3 and ArcGIS to cope with VROs in public utilities. Experiments on a real-world case in Bogotá, Colombia with 323–601 customers verified the effectiveness of evolutionary optimization and GIS software.

Recently, progress of GIS related to cloud computing transfers the integration of GIS and VRO to a loose coupling way. Using Google maps, Santos et al. [47] developed user-friendly web-based SDSSs that embed VRO to deal with urban trash collection in Coimbra, Portugal. TU et al. [48] presented a cloud GIS-based spatial decision support framework with variable neighborhood search heuristics for dynamic vehicle routing using historical traffic information. These practices have proven the dominance of VROs in real-life transportation applications. However, they also indicate that current spatial intelligence should be improved to enhance the ability of SDSSs in transportation departments to cope with increasing numbers of customers.

In summary, state-of-the-art heuristics, including local search and evolutionary optimization, solve small- and medium-size VRPTWs in a reasonable time. They have been verified to be effective in a GIS environment for real-life transportation cases with no more than 1000 customers. For larger-size real-life applications, more efficient heuristic algorithms are needed to reduce computing time. Using the spatial-temporal Voronoi diagram, this paper proposes an efficient local search-based approach to address VRPTWs up to 10,000 customers in a reasonable time. Differing from the neighborhood in the spatial context [25,26,27,48,49], the spatial-temporal Voronoi neighborhood considering both spatial distance and time windows is proposed to limit the search space. A Voronoi distance decay strategy is developed to assign more searching efforts to nearer neighbors. The spatial-temporal feature guided search is used to escape from local minima. Details of the STVD, Voronoi

distance, Voronoi neighbors, and the proposed approach for the VRPTW are described in the next sections.

3. The VRPTW and the Spatial-Temporal Voronoi Diagram

This section introduces the VRPTW, the STVD, and the spatial-temporal Voronoi distance.

3.1. The VRPTW

The VRPTW has five components: customers, depot, vehicle, routes, and the solution.

Customers: a set of customers $V_c = \{v_1, v_2, \ldots, v_n\}$ expects to be served. A customer v_i (I > 0) located at (x_i, y_i) has a unique demand, q_i, a service duration, s_i, and a service time window <e_i, l_i>. The service is required to arrive after start time, e_i, and before end time, l_i. Such a time window is termed "soft" when it could be violated with a penalty cost, and termed "hard" when no violation is permitted.

Depot: The depot v_0 located at (x_0, y_0) provides service in a time window <e_0, l_0>.

Vehicles: A fleet of K identical vehicles with the capacity Q is located at the depot. Each vehicle departs from the depot, serves multiple customers, and returns to the depot. Travel distance and time between nodes (includes customers and the depot) i, j are denoted as d_{ij} and t_{ij}, respectively, where $i \neq j$.

Routes: A route $r = \langle v_0, v_1, v_2, \ldots, v_R, v_{R+1} \rangle$ is formed by a sequence of visited customers, which represents the service process of a vehicle, where R denotes the number of visited customers. , so that the vehicle must return to the depot. The $v_{R+1} = v_0$ earliest departure time at the depot is a_{v_0}, the earliest service time at a customer is a_{v_i} and the earliest return time at the depot is $a_{v_{R+1}}$ which are defined recursively as follows:

$$\begin{cases} a_{v_0} = a_r \\ a_{v_i} = \max\{a_{v_{i-1}} + s_{v_{i-1}} + t_{i-1,i}, e_{v_i}\}, (i = 1, \ldots, R+1) \end{cases} \quad (1)$$

A route is specified following two restrictions: (1) The route duration time should be no more than the maximum duration T_{max}; (2) The total served demand $a_{v_{R+1}} - a_r$ should be no more than the vehicle capacity Q. The total travel distance of the route is defined as $\sum_{i=1}^{R} q_i$. A route is defined as feasible if the customer's service time window, the route duration time, and the vehicle capacity are satisfied. $\sum_{i=0}^{R} d_{v_i v_{i+1}}$

The VRPTW solution: A solution for a VRPTW instance is a set of feasible routes $r_1, r_2, r_3, \ldots, r_K$ such that a single vehicle visits each customer once. A hierarchy of objectives of the VRPTW is followed in this paper. The number of used vehicles is first minimized and then the total travel distance is reduced.

3.2. The Spatial-Temporal Voronoi Diagram for VRPTWs

Components of the VRPTW, such as customers, depot, vehicles, and routes, have both spatial and temporal attributes. Spatial attributes refer to customers' locations, the depot location, and the vehicle trajectories.

Temporal attributes refer to the time window, service time, travel time, and route duration time. To represent them simultaneously, a spatial-temporal coordinate system is built by taking the temporal dimension to be perpendicular to the planar XY plane. Therefore, time windows of the customers and the depot are modeled as vertical lines rooted in their locations.

illustrates a small-size VRPTW with five customers (v_1 to v_5), one depot (v_0), and two routes (R1, R2). Customers (v_1–v_5) and the depot (v_0) are modeled as vertical lines from its start time point to end time point, as line segments L_1–L_5 and L_0. Projections on the XY plane denote their spatial locations, whereas projections on the time dimension denote their time windows and actual service time. The transportation between customers denotes the service process in the routes (R1 or R2).

Distance in this coordinate system is defined as Equation (2), where (x_i, y_i) and (x_j, y_j) denote the spatial location of i and j, and t_i and t_j denote the time at i and j, $\bar{v}$s the mean travel speed:

$$d(v_i, v_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (\bar{v}(t_i - t_j))^2} \quad (2)$$

Voronoi diagrams partition space into adjacent Voronoi cells with given points [30]. Following this principle, a spatial-temporal Voronoi diagram divides the spatial-temporal space into many cells. To generate a Voronoi diagram for the VRPTW, the spatial-temporal central point of a customer's spatial temporal line, (x_i, y_i, (e_i + l_i)/2.0), which is the red point in , is used as the seed point. The STVD for VRPTWs is built by the fast quickhull algorithm [50].

In the STVD, a pair of points whose cells share a boundary (Voronoi face or Voronoi edge) are Voronoi neighbors [30]. The spatial-temporal Voronoi distance from a given point i to the point j, vd(i, j), is defined as the minimum number of crossings of the spatial-temporal Voronoi boundary in any route from i to j [27,31]. provides an example that illustrates the Voronoi distance. The Voronoi distance from P1 to P3 is 2 for the two crossings of Voronoi boundaries (C1–C2). The Voronoi distance from P1 to P13 is also 2 for the crossings C3–C4. The Voronoi distance from P1 to P12 is 3 for the crossing C5–C7. Thus, Voronoi neighbors with different spatial-temporal proximity can be measured. As both spatial distance and time windows are considered, it is well suited for accelerating local search for the solving of VRPTW. Using these useful definitions, this paper proposes an efficient local search heuristic to address large-scale VRPTWs.

Figure 1. Spatial-temporal representation of VRPTW.
Figure 1. Spatial-temporal representation of VRPTW.

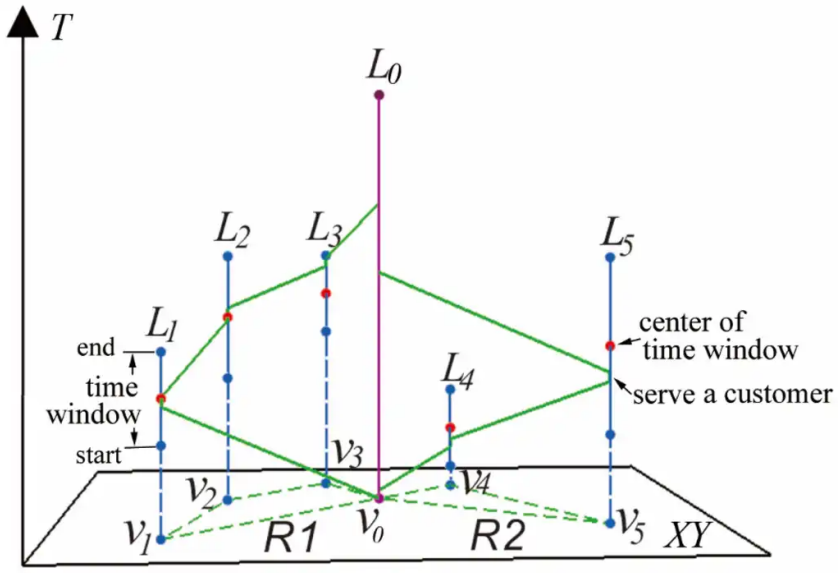
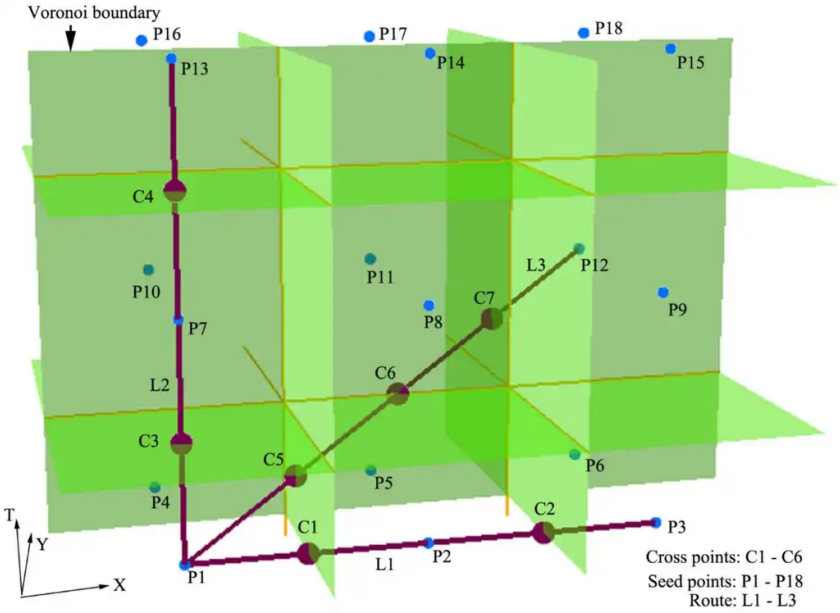


Figure 2. The spatial-temporal Voronoi diagram and the spatial-temporal Voronoi distance.

Figure 2. The spatial-temporal Voronoi diagram and the spatial-temporal Voronoi distance.



4. The Spatial-Temporal Voronoi Diagram-Based Heuristic

This section introduces the spatial-temporal Voronoi diagram-based heuristic (STVDH) for large-scale VRPTWs. The principle of the STVDH is to speed up the local search procedure by using useful spatial thinking, including the spatial-temporal Voronoi neighbors, distance decay search strategy, and local route features. The overall framework of the proposed heuristic is summarized in . After the construction of an initial solution (step 1-1), a local search heuristic based on spatial-temporal Voronoi neighborhood is followed to improve solution quality (step 1-1 to step 1-4). A spatial-temporal Voronoi distance decay strategy is developed to assign reasonable searching efforts on neighbors with different distances (step 1-2). Spatial-temporal route features are identified and guides the searching process escape from local minima (step 1-3). The stopping condition is a limitation on the number of iterations (step 1-4). Finally, the best found solution is reported (step 1-5).

Figure 3. The framework of the spatial-temporal Voronoi diagram-based heuristic algorithm.

Figure 3. The framework of the spatial-temporal Voronoi diagram-based heuristic algorithm.

[The spatial-temporal Voronoi diagram based heuristic (STVDH)]

Input: VRPTW instance,
parameters setting:
N_{max}: maximum number of iterations
Z_{max} : maximum number of non-improved iterations to start spatial-temporal guided local search
$k_{\max}$:maximum local search Voronoi distance
α : parameter controlling the distance decay speed.

Output: the found best solution S_{best}

Step 1-0: Read VRPTW instance and build the spatial-temporal Voronoi diagram with customer nodes.

Step 1-1: Generate an initial solution S_0 with the construction algorithm in section 4.1. Set $S_{best} = S_0$.

Step 1-2: Explore the solution neighborhood by the local search in section 4.2. Update S_{best} if a better solution is found. If the solution quality isn't improved in Z_{max} iterations, record the local minima S_{local}, go to step 1-3; else, repeat this step.

Step 1-3: Implement the spatial-temporal feature guided local search in section 4.3 to disturb S_{local}. Spatial-temporal features in S_{local} are identified and a modification process is followed.

Step 1-4: Check the stopping criteria with N_{max}. If the stopping criterion is accepted, go to step 1-5. Otherwise, go to step 1-2.

Step 1-5: report S_{best} and exit.

4.1. The Construction Algorithm

The construction algorithm iteratively inserts an unrouted node into a proper position until all customers are routed. During the insertion, instead of evaluating all unrouted nodes, neighbor customers within a given Voronoi distance are selected as candidates of the next inserted nodes. The details of the construction algorithm are described in . First, customers are sorted by the start time window and stored in a queue L (step 2-0). Then, L's front unrouted node is popped out as the operated node (step 2-1). An empty route r is initialized with u (step 2-2). Another unrouted customer v is randomly selected from μ Voronoi neighbors. Node v will be inserted before or after a routed node i in the route r (step 2-3). If the insertion is performed, u is updated with v. The best insertion place is defined as the one that minimizes the added total travel length and the total value of the end time window violation as Equation (3), where j denotes the next visited node of i. λ is a random value in [0.5, 2]. It should be noted that the arrival time at v, j should not be before the start time, as Equations (4) and (5):

$$\Delta F = (d_{iv} + d_{vj} - \lambda d_{ij}) + \bar{v}\,(a_i + s_i + t_{iv} - l_v) \quad (3)$$

$$a_i + s_i + t_{iv} > e_v \quad (4)$$

$$a_i + s_i + t_{iv} + s_v + t_{vj} > e_j \quad (5)$$

Figure 4. The construction algorithm.

Figure 4. The construction algorithm.

[Construction algorithm]

Step 2-0: Label all customers as unrouted. Sort customers with start time window and store them in a queue *L*. Initialize an empty route *r*.

Step 2-1: **If** *L* is empty, go to step 2-4;
Else,
pop *L*'s front node *u*.
If *u* is unrouted, go to step 2-2;
Else, repeat step 2-1.

Step 2-2: Insert node *u* into an empty route *r* and label *u* as routed.

Step 2-3: Select an unrouted node *v* randomly from Voronoi neighbors of the last inserted node *u*. Find the best insertion place *i*, insert *v* before or after *i*, set *u*=*v*, and label *v* as routed. Repeat this step until there is no place for insertion, go to step 2-1.

Step 2-4: If all customers are routed, report solution and exit.

During the insertion, several constraints including the start time window, route duration time, vehicle capacity, and the maximum route length should be satisfied. If no place is found for insertion, we pop L's front node as u, insert it into a new route, and label it as routed (step 2-2 and step 2-3). The insertion will end if all customers have been labeled as routed (step 2-5). Finally, obtaining all routes is the initial solution.

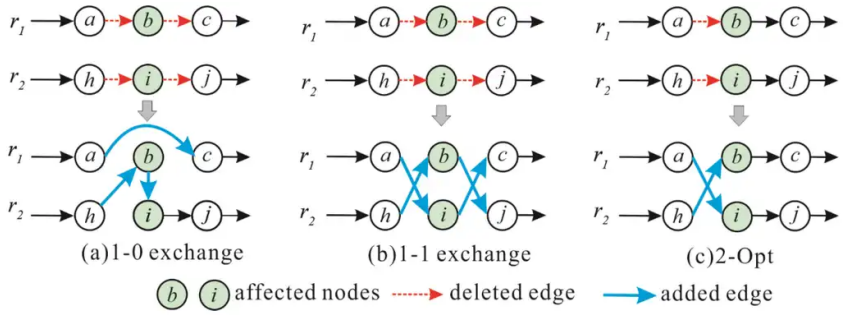
4.2. Local Search

Local search iteratively explores the solution space to improve solution quality. Usually, the exploration removes or adds edges from the current solution by using local search operators. Three typical local search operators, including the 1-0 exchange move, 1-1 exchange move, and 2-Opt move [51], are used to explore the neighborhood solution in this article.

- 1-0 exchange move injects a node from the original place and inserts it after another. As a shows, node b is removed from its current position and inserted before node i.
- 1-1 exchange move swaps positions of a pair of nodes. As b shows, the places of nodes b and i are swapped.
- 2-Opt move swaps the ends of two routes after the positions of a pair of nodes. As c illustrates, the partial routes after b and i are swapped.

Figure 5. Local search operators (**a**) 1-0 exchange; (**b**) 1-1 exchange; and (**c**) 2-Opt.

Figure 5. Local search operators (**a**) 1-0 exchange; (**b**) 1-1 exchange; and (**c**) 2-Opt.



The computing complexity of these three operators depends on two aspects: operated nodes and local search evaluation. Two operated nodes, named b and i, are involved as shown in . Therefore, the number of operated candidates is n(n − 1), where n is the number of customers. The

time complexity is . For each candidate pair, local search evaluation checks the time window for each customer involved, total duration time, and vehicle capacity for each route. Due to the cascade effect of the arrival time, it will, on average, take time, where R is the number of routes. To utilize limited computing effort more efficiently, we propose a spatial-temporal Voronoi distance-decay strategy to assign searching efforts on selected candidates. A time warp operation is also used to shorten computing time for each candidate.

$O(n^2)$

$O(n/2R)$

4.2.1. The Spatial-Temporal Distance Decay Strategy

Differing from the equal searching intensity of the k nearest neighbors [26] and the k-ring Voronoi neighbors [27,48], the spatial-temporal Voronoi distance decay strategy searches more on nearer neighbors than further neighbors to balance the solution quality and computing time. For spatial-temporal Voronoi neighbors with Voronoi distance k, the searching probability P_k is defined as Equation (6). As the distance k increases, searching probability P_k will decrease. Therefore, most computing effort will focus on near neighbors, but some necessary efforts are left for further neighbors.

$$P_k = \frac{\alpha^{-k}}{\sum_{k=1}^{k_{max}} \alpha^{-k}} \quad (6)$$

where k = 1, …, k_{max}, and k_{max} is the maximum searchable Voronoi distance, is a value that controls the spatial-temporal distance decay speed. As the value of increases, the decay speed becomes faster.

$\alpha \in [1, \infty)$

α

The probability range $[TP_{k-1}, TP_k]$ (k = 1, 2, …, k_{max}) for neighbors no farther than Voronoi distance k is calculated as Equation (7), where $TP_0 = 0$.

$$TP_k = \sum_{i=1}^{k} P_i \quad (7)$$

To implement the Voronoi distance decay strategy, local search randomly generates a value of between (0, 1), finds the probability range in which is located, and then selects Voronoi neighbors with the corresponding distance k as candidates to evaluate. As it avoids evaluating all neighbors, this useful strategy puts more searching efforts on closer nodes and therefore significantly accelerates the local search. As the number of Voronoi neighbors is fixed, the computing complexity reduces from to .

δ

δ

$O(n^2)$ $O(n)$

4.2.2. Time Warp Operation

The local search evaluates distance and time constraints for customers and routes, which cost the most computing effort [43]. Nagata [45] proposed a new penalty function in which change can be computed in O(1) time for the VRPTW. When a later service happens for a customer, a time warp operation is performed such that the arrival time is adjusted to the end of the time window, and a penalty is added to the original objective. shows an example of the time warp. The additional penalized cost is defined as the time warp used in Equations (8) and (9), where tw_i is the cost value at customer i, and $(a_i - e_i)^+$ is a non-negative value of the service late time at i. Therefore, the objective is changed to minimize the total route number,

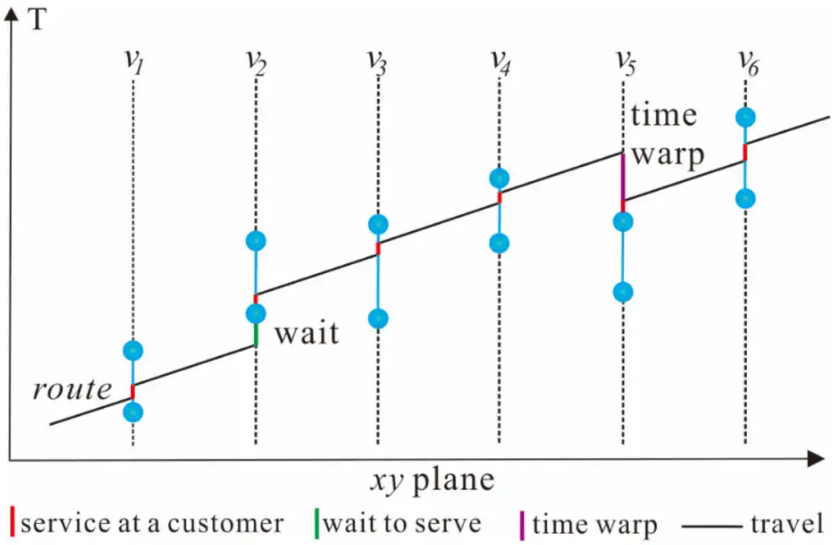
total route length, and total penalized cost TW(s). For O(1) time consumed, the time warp operation is very fast.

$$TW(s) = \sum_{i=1}^{n} tw_i \quad (8)$$

$$tw_i = (a_i - e_i)^+ \quad (9)$$

Figure 6. Wait time and time warp at a customer.

Figure 6. Wait time and time warp at a customer.



4.2.3. Acceptance Criterion

Local search will naturally drop into local minima. A threshold acceptance criterion is used to escape. If the found solution is not 0.5% worse than the current solution, it will be accepted as the new solution. Otherwise, it will be rejected. If the best solution does not improve in Z_{max} iterations, the spatial-temporal features-guided search will be started.

4.3. Spatial-Temporal Features-Guided Search

Due to the used spatial-temporal Voronoi neighbors, local search explores a relatively small proportion of the neighborhood solutions. Some potential neighborhood solutions will be rejected due to spatial and temporal constraints, such as route duration, time window, and vehicle capacity. Different from traditional random perturbations [37], spatial-temporal features guided search destroys unpromising structures defined by spatial-temporal rules and rebuilds a part of the current solution to escape local minima. It also increases the searchable probability for farther neighborhood solutions. Three components are the spatial-temporal features, the removal algorithm, and the reinsertion algorithm.

4.3.1. Spatial-Temporal Features

Three types of spatial-temporal features are defined in this paper including time window violation nodes φ_t, longest distance nodes φ_d, and smallest route's nodes φ_s. Their definitions are described below.

- Time-window violation nodes

The time warp operation will introduce time window lateness in which the arrival time at a customer may be later than the end time of its time window. However, this route feature should not belong to the best solution. We remove such unpromising nodes from the current solution and reinsert them later. Formally, time window violation nodes φ_t are defined as Equation (10):

$$\varphi_t = \{v_i | t_i > l_i, v_i \in V_c\} \tag{10}$$

- Longest-distance nodes

To cooperate with the time window, some long segments may be kept in routes and difficult to replace in local search, which leads to the increase of total route length [27]. Longest-distance nodes φ_l are end points of the long segment, defined as Equation (11), where S denotes the current solution, e_{ij} or e_{ji} denotes a segment in S. $\bar{e}$ denotes the mean spatial distance of the Voronoi neighbors with Voronoi distance 1. The length of the long segment is more than three times $\bar{e}$

$$\varphi_l = \{v_i | d_{ij} > 3\bar{e}, e_{ij} \in S || e_{ji} \in S, v_i \in V_c\} \tag{11}$$

- The smallest route's nodes

In some solutions, small routes serve only one, two, or three customers near the depot, which requires more vehicles [48]. Usually, this feature is kept for the suboptimal property. The small route's nodes are defined as φ_s as Equation (12), where |r| denotes the number of visited customers in the route r:

$$\varphi_l = \{v_i | v_i \in r, |r| \leq 3, v_i \in V_c\} \tag{12}$$

4.3.2. Removal Algorithm

To increase diversification of the guided search, more nodes are removed from the current solution in this study. Additional nodes φ_a are Voronoi neighbors of defined features, just as:

$$\varphi_a = \{v_i | vd(v_i, v_j) = 1, v_j \in \left(\varphi_t \cup \varphi_l \cup \varphi_s\right), v_i \notin \left(\varphi_t \cup \varphi_l \cup \varphi_s\right), v_i \in V_c\} \tag{13}$$

Therefore, removed nodes φ are spatial-temporal features nodes and their Voronoi neighbors φ_a as in Equation (14). The removal algorithm sequentially ejects all nodes belonging to φ and leaves a partial solution for the insertion.

$$\varphi_a = \left\{ v_i \middle| v_i \in \left(\varphi_t \cup \varphi_l \cup \varphi_s \cup \varphi_a\right), v_i \in V_c \right\} \tag{14}$$

4.3.3. Insertion Algorithm

This algorithm inserts all removed nodes into new places in the partial solution. First, all removed nodes are randomly pushed into a queue L. Then, the front node of L is popped out and inserted at the best place before or after its Voronoi neighbors to keep spatial-temporal proximity, which minimizes insertion cost as Equation (3), under the limitation of total duration time, time window, and vehicle capacity. It is of note that the end time windows should not be violated. If there is no proper place, a new

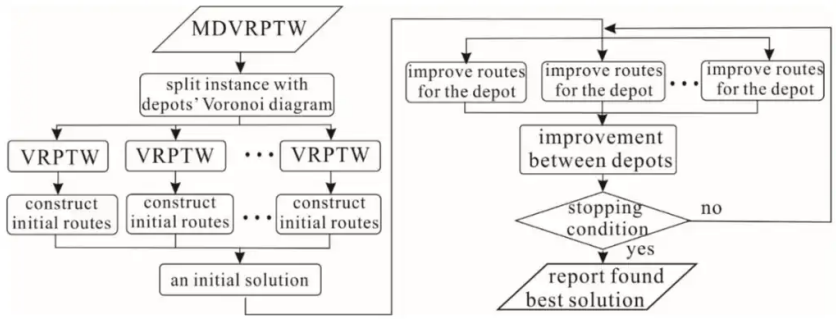
route is created, and the front node of L is popped out and initialized as the first-served customer. The insertion operation is repeated until all removed nodes are located in proper places. Due to the best choice at each insertion, the final solution is up to the node sequence in the queue. In this article, the sequence is randomly arranged 50 times to obtain different rebuild solutions. Finally, the obtained result of the permutation that minimizes the total objective is accepted as the final solution.

4.4. An Extension of the Solving Algorithm for MDVRPTW

MDVRPTW is a variant of VRPTW, which has more than one depot to serve geographically scattered customers. Compared with the VRPTW, MDVRPTW has an additional problem of how to allocate customers to the right depot. Voronoi partition that divides a study area into many regions with given objects is another typical function of the Voronoi diagram. For the effectiveness, it has been widely used in service area analysis [32]. By using Voronoi partition characteristics of the depots' spatial-temporal Voronoi diagram, we assign customers to be served by the depot of its located Voronoi regions. By that, the presented STVDH is easy to be extended to deal with a large-scale MDVRPTW by decomposing it to many smaller VRPTWs.

illustrates the workflow of the STVDH's adaption for the MDVRPTW. The depots' spatial-temporal Voronoi diagram is built to allocate customers to the located depot so that the MDVRPTW is divided into several VRPTWs, each of which has a depot and many assigned customers. Routes for each VRPTW are first initialized using the construction algorithm in Section 4.1 and then separately improved by the spatial-temporal Voronoi diagram-based local search heuristic in Section 4.2 and Section 4.3. To overcome the border effect of Voronoi diagram, an additional improvement between depots is used to move or exchange nodes between routes served by different depots. This is done by using 1-0 and 1-1 exchanges to move nodes between routes served by different depots after the spatial-temporal features guided search. Details of this process are referred to TU et al. [49]. The stopping condition is the same as the STDVH. Finally, the best found solution is reported.

Figure 7. The adaption of the STVDH for the MDVRPTW.
Figure 7. The adaption of the STVDH for the MDVRPTW.



5. Experiment and Comparison

To assess the performance of the STVDH, experiments with benchmark problems and large-scale real-world VRPTW and MDVRPTW problems in Shanghai, China were conducted. The STDVH algorithm was implemented

with C++, running on a Windows 7 64-bit system with an Intel Core i7-3.4G processor and 16 GB memory. This section reports test datasets, parameter tuning, computing results, and comparisons with other heuristic algorithms. Computing times of compared algorithms are transformed to the computing platform of STVDH with Dongarra factor [52].

5.1. Test VRPTW and MDVRPTW Problem Dataset

Two types of VRPTW and MDVRPTW problem datasets are tested. The first dataset includes the VRPTW benchmarks of Solomon [15] and Gehring and Homberger [53]. The benchmarks of Solomon have 56 problems with 100 customers. They are divided into six groups with eight to twelve problems, named R1, R2, C1, C2, RC1, and RC2. In the Euclidean plane, customers are randomly distributed (R1 and R2), cluster distributed (C1 and C2) or semi-cluster distributed (RC1 and RC2). Every instance has a single depot within the spatial domain of the customers. The travel time between nodes is equal to the corresponding Euclidean distance. R1, C1, and RC1 have a short time scheduling horizon, whereas R2, C2 and RC2 have a long time scheduling horizon. Similar to Solomon's VRPTW problems, the VRPTW benchmarks of Gehring and Homberger [53] have 300 VRPTW instances with 200, 400, 600, 800 or 1000 customers [54]. Customers have the same distributions.

The second dataset is a large-scale VRPTW and MDVRPTW problem dataset in Shanghai, China [55]. Five VRPTW instances (Sh1a–Sh5a) and 5 MDVRPTW instances (Sh1b–Sh5b) with 2000 to 10,000 customers and one to six depots are generated to simulate the daily parcel delivery service of a logistics company in Shanghai, China. Customers are randomly selected from the commercial points of interest (POI) using a professional digital navigation map to obtain real-world spatial locations. The time window of each customer is a random value in min bounded by 30 and 60, and the service duration is a random value from 1 to 4 min. Demand is a random value in the interval [1, 100]. Depots are located in professional logistic districts in the city. All vehicles have the same capacity, 2000, and maximum route duration, 480 min. The mean travel speed is set to be 45 km/h.

5.2. Parameter Tuning

There are four parameters to be tuned in the presented STVDH algorithm as shown in . Following Coy's calibration approach [56], a preliminary experiment is conducted on an instance in the problem dataset to find the best parameter settings. The tuning process initially identifies the first two parameters related to the stopping condition and then the last two parameters related to the Voronoi distance decay strategy. The first two parameters, the maximum number of iterations, N_{max}, and the maximum number of iterations to start the spatial-temporal features guided local search, Z_{max}, control the exit of the STVDH. After intensive experiments with different stopping conditions, N_{max} should be set to $5000\mathbf{N}$ to converge on a stable high-quality solution, where $\mathbf{N}$ is the number of customers. Z_{max} is set to $100\mathbf{N}$ to start the spatial-temporal feature guided search.

The last two parameters control the Voronoi diagram-based speedup strategy. The third parameter, k_{max}, indicates the maximum searchable Voronoi neighbors. As the value of k_{max} increases, local search will evaluate more neighbors, which requires more computing efforts. According to the distribution of the Voronoi neighbors, k_{max} should be no more than 5 [49]. The last parameter,, determines the distance decaying speed. A high value indicates a tendency for fast decay. The proper value of α is within the bound [1,4]. The value space of parameters are summarized in .

Table 1. Parameter settings of the STDVH algorithm.

Table 1. Parameter settings of the STDVH algorithm.

Notation	Parameter	Setting
N_{max}	the maximum number of iterations	5000**N**
Z_{max}	the maximum number of iterations to start the spatial-temporal features guided search	100**N**
k_{max}	the maximum searching Voronoi neighbors	1, 2, 3, 4, 5
α	Voronoi distance decaying speed	(0, 4)

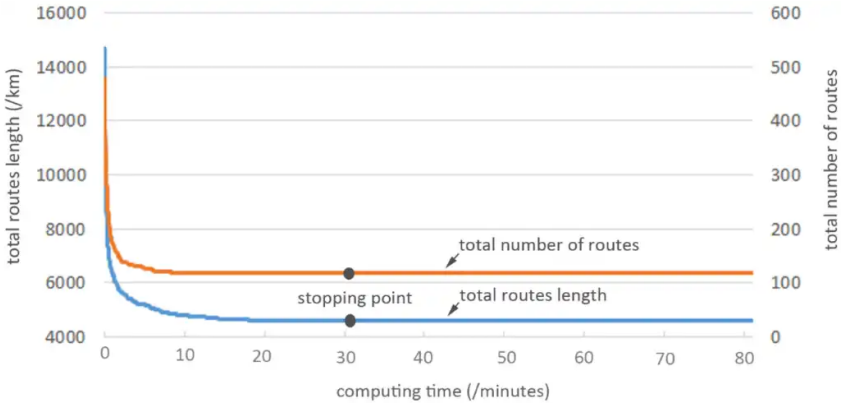
Experiments on some selected VRPTW instances are conducted to set parameter values for one type of VRPTW dataset. Taking the second VRPTW dataset as an example, we solved Sh2a with all combinations of the last two parameters. Finally, we set k_{max} = 2 and = 2 for the best performance in the experiment. Without loss of generality, identical parameter values were set for each instance. The impact of the Voronoi diagram will be discussed in Section 6. A summary of parameter settings is listed in . Taking Sh2a as an example, displays the convergence of the number of routes and the total length of routes with the parameter settings used in solving Sh2a.

5.3. The Results of the Solomon [15] and Gehring and Homberger [53] VRPTW Benchmarks

The parameter setting for these VRPTW benchmarks is as: N_{max} = 5000**N**, Z_{max} = 100**N**, k_{max} = 2 and α = 1.5. For each problem, the STDVH was run 10 times. The total objective, total routes length, and computing time were recorded. Results of three state-of-the-art methods, including the arc-guided evolutionary algorithm (AGEA) [39], the hybrid genetic algorithm with adaptive diversity management (HGASDC) [40], and the local search based heuristics VRPEJ [57], which limits the searching space with the kth nearest neighbor strategy, are compared with the STVDH.

Figure 8. The convergence of the total number of routes and the total routes length (Sh2a).

Figure 8. The convergence of the total number of routes and the total routes length (Sh2a).



and present the comparison of the results on the Solomom and Gehring and Homberger VRPTW benchmarks respectively. Each entry reports the average of computing results of a group (average computing time | the average total routes length). They indicate that the presented STVDH is able to achieve high quality solutions of the VRPTWs with no more than 1000 customers in a relatively short computing time. In terms of the number of routes, the STVDH achieves the fewest routes in both the Solomon (total of 405 routes) and Gehring and Homberger (total of 10296 routes) VRPTW benchmarks. In terms of the total route length, the STVDH performs better than the VRPEJ for all size VRPTW instances, only inferiorly to AGEA on Solomon benchmarks, but superiorly to AGEA on the Gehring and Homberger benchmarks. However, the STVDH's results are still worse than that of the HGSADC. For the computing efficiency, the STVDH consumes the least computing time in all six size VRPTW instances (100~1000). Therefore, the proposed STDVH exhibits a good performance on the VRPTW benchmarks of Solomon and Gehring and Homberger.

Table 2. Comparison of results on the Solomon VRPTW benchmarks [15].

Table 2. Comparison of results on the Solomon VRPTW benchmarks [15].

Instance	N	AGEA 1 run	HGSDAC Best 5 run	VRPEJ Best 10 run	STVDH AVG 10 run	STVDH Best 10 run
R1	100	11.92|1211.43	11.92|1210.69	11.92|1214.67	11.92|1213.85	11.92|1212.65
R2	100	2.73|954.05	2.73|951.51	2.73|954.10	2.73|954.01	2.73|953.20
C1	100	10.0|828.38	10.0|828.38	10.0|828.38	10.0|828.38	10.0|828.38
C2	100	3.0|589.86	3.0|589.36	3.0|589.86	3.0|589.86	3.0|589.86
RC1	100	11.50|1384.83	11.5|1384.17	11.5|1387.81	11.5|1386.93	11.5|1386.01
RC2	100	3.25|1121.26	3.25|1119.24	3.25|1127.38	3.25|1126.49	3.25|1125.72
CNV		**405**	**405**	**405**	**405**	**405**
CTD		**57,254.73**	**57,195**	**57,366.96**	**57,341.97**	**57,305.10**
T(/min)		**180**	**2.68**	**3.62**	**0.91**	**0.9**
T2(/min)		**52.3**	**1.56**	**3.62**	**0.91**	**0.9**

Table 3. Comparison of results on the Gehring and Homberger VRPTW benchmarks [53].

Table 3. Comparison of results on the Gehring and Homberger VRPTW benchmarks [53].

Instance	N	AGEA 1 run	HGSDAC Best 5 run	VRPEJ Best 10 run	STVDH AVG 10 run	STVDH Best 10 run
R1	200	18.2|3640.11	18.2|3613.16	18.2|3664.28	18.2|3653.24	18.2|3617.54
R2	200	4.0|2941.99	4.0|2929.41	4.0|2938.53	4.0|2967.87	4.0|2938.53
C1	200	18.9|2721.90	18.9|2718.41	18.9|2749.18	18.9|2758.67	18.9|2721.47
C2	200	6.0|1833.36	6.0|1831.59	6.0|1880.47	6.0|1858.34	6.0|1835.52
RC1	200	18.0|3224.63	18.0|3180.48	18.0|3205.81	18.0|3218.83	18.0|3179.89

Instance	N	AGEA 1 run	HGSDAC Best 5 run	VRPEJ Best 10 run	STVDH AVG 10 run	STVDH Best 10 run
RC2	200	4.3|2554.33	4.3|2536.20	4.3|2574.92	4.3|2584.37	4.3|2544.44
CNV		**694**	**694**	**694**	**694**	**694**
CTD		**169,163**	**168,092**	**176,440.8**	**170,415.5**	**168,373.7**
T(/min)		**90**	**8.40**	**5.4**	**1.22**	**1.2**
T2(/min)		**26.7**	**4.9**	**5.4**	**1.22**	**1.2**
R1	400	36.4|8514.11	36.4|8402.57	36.4|8615.29	36.4|8598.39	36.4|8446.36
R2	400	8.0|6258.82	8.0|6152.92	8.0|6274.20	8.0|6279.13	8.0|6160.84
C1	400	37.6|7273.90	37.6|7170.47	37.6|7339.88	37.6|7514.01	37.6|7186.10
C2	400	11.7|3941.70	11.6|3950.95	11.7|4024.82	11.7|3995.21	11.7|3951.71
RC1	400	36.0|8088.46	36.0|7907.14	36.0|8107.86	36.0|8098.32	36.0|7952.00
RC2	400	8.40|5516.59	8.5|5215.21	8.4|5394.54	8.4|5393.49	8.4|5292.92
CNV		**1381**	**1381**	**1381**	**1381**	**1381**
CTD		**395,936**	**388,013**	**397,565.9**	**398,785.5**	**389,385.5**
T(/min)		**180**	**34.1**	**9.8**	**2.30**	**2.3**
T2(/min)		**53.3**	**19.8**	**9.8**	**2.30**	**2.3**
R1	600	54.5|18,781.79	54.5|18,023.18	54.5|18,620.73	54.5|18,587.90	54.5|18,237.74
R2	600	11.0|12,804.60	11.0|12,352.38	11.0|12,615.07	11.0|12,612.60	11.0|12,343.51
C1	600	57.3|14,236.86	57.4|14,058.46	57.3|14,605.53	57.3|14,585.55	57.3|14,271.58
C2	600	17.4|7729.80	17.4|7594.41	17.4|7748.47	17.4|7728.74	17.4|7589.10
RC1	600	55.0|16,767.72	55.0|16,097.05	55.0|16,529.63	55.0|16,524.77	55.0|16,203.93
RC2	600	11.4|11,311.81	11.5|10,511.86	11.4|10,879.26	11.4|10,824.00	11.4|10,626.35
CNV		**2066**	**2068**	**2068**	**2066**	**2066**
CTD		**816,326**	**786,793**	**809,986.9**	**808,635.65**	**792,722.1**
T(/min)		**270**	**99.4**	**16.2**	**3.91**	**3.9**
T2(/min)		**80.0**	**57.8**	**16.2**	**3.91**	**3.9**
R1	800	72.8|32,734.57	72.8|31,311.38	72.8|32,281.48	72.8|32,108.01	72.8|31,540.28
R2	800	15.0|20,618.21	15.0|19,933.39	15.0|20,448.88	15.0|20,339.05	15.0|19,969.61
C1	800	75.2|25,911.44	75.4|24,876.93	75.2|26,097.53	75.1|25,972.63	75.1|25,490.85
C2	800	23.4|11,835.72	23.3|11,475.05	23.4|11,897.31	23.3|11,826.41	23.3|11,621.87
RC1	800	72.0|33,975.61	72.0|29,404.32	72.0|31,071.16	72.0|30,904.01	72.0|30,390.41
RC2	800	15.5|17,536.54	15.4|16,495.82	15.5|16,878.69	15.4|16,733.78	15.4|16,467.01
CNV		**2739**	**2739**	**2739**	**2736**	**2736**
CTD		**1,424,321**	**1,334,963**	**1,386,750.4**	**1,378,838.8**	**1,354,800**
T(/min)		**360**	**215**	**24.8**	**5.82**	**5.8**
T2(/min)		**106.6**	**125.1**	**24.8**	**5.82**	**5.8**
R1		91.9|51,414.26	91.9|47,759.66	91.9|49,741.43	91.9|49,396.91	91.9|48,523.49
R2		19.0|30,804.79	19.0|29,076.45	19.0|29,871.68	19.0|29,595.30	19.0|29,092.01
C1		94.2|43,111.60	94.1|41,572.86	94.1|43,089.45	94.1|42,682.27	94.1|41,977.06
C2		29.3|16,810.22	28.8|16,796.45	29.0|117,340.13	28.9|17,174.73	28.8|16,877.68
RC1		90.0|46,753.61	90.0|44,333.40	90.0|46,152.97	90.0|45,824.65	90.0|44,974.63
RC2		18.4|25,588.52	18.2|24,131.12	18.4|24,951.31	18.3|24,655.48	18.2|24,248.11
CNV		**3428**	**3420**	**3421**	**3419**	**3419**
CTD		**2,144,830**	**2,036,700**	**2,111,469.7**	**2,093,293.5**	**2,056,930**
T(/min)		**450**	**349**	**34.5**	**7.75**	**7.7**
T2(/min)		**133.3**	**203.1**	**34.5**	**7.75**	**7.7**

5.4. The Results of Large Scale VRPTW and MDVRPTW Problem in Shanghai, China

The parameter setting for this large scale VRPTW and MDVRPTW problem is as follow: N_{max} = 5000**N**, Z_{max} = 100**N**, k_{max} = 2 and α = 2. For each problem, the STVDH algorithm was also run 10 times. Total travel distance, number of used vehicles, and computing time were recorded. The obtained results are summarized in (https://github.com/spatialsmart/VRPTW/tree/master/Shanghai/Solution).

As indicates, the STVDH algorithm reports the best found solution for large-scale VRPTWs up to 10,000 customers in 139.5 min (about 2.32 hours). As the number of customers increases from 2000 to 10,000, the computing time of the STVDH increases from 15.7 min to 120.8 min, whereas for the MDVRPTW, the time increases from 16.1 min to 139.5 min. Compared with the VRPTW, more computing efforts are required for the MDVRPTW due to additional customer allocation between depots. In terms of solution quality, the STVDH utilizes an average of 847.1 total routes that travelled 30,184.725 km to serve all customers in the five VRPTW instances. For the five MDVRPTW instances, 906.4 routes that travelled 26,353.572 km are required to satisfy customer needs. It should be noted that the performance of the STVDH is robust, as indicated by the gap between S_{avg} and S_{best} (-1.60% for the VRPTW and -1.73% for the MDVRPTW). Therefore, the proposed STVDH provides promising results for large-scale VRPTW and VRPTW within reasonable computing times.

Taking problem Sh1a as an example, displays the best found solution and the spatial-temporal vehicle route with ArcScene. It demonstrates the best route compromise between spatial proximity and time constraints. The same vehicle may not yet serve spatially-near customers as b shows.

To assess the solution quality of the obtained results, we compare them with solutions reported by two heuristic algorithms. One solution is obtained by solving the simulated instances with the network analysis module in ArcGIS^{TM} 10.2, which uses a tabu heuristics algorithm to solve complex real-world VRPTW problems. Another compared solution is calculated using VRPEJ [55]. Other metaheuristics such as AGEA and HGSDAC are not compared due to unavailability. Each algorithm was run 10 times to find the best result.

Table 4. Results from the large-scale VRPTW and MDVRPTW instances in Shanghai, China.

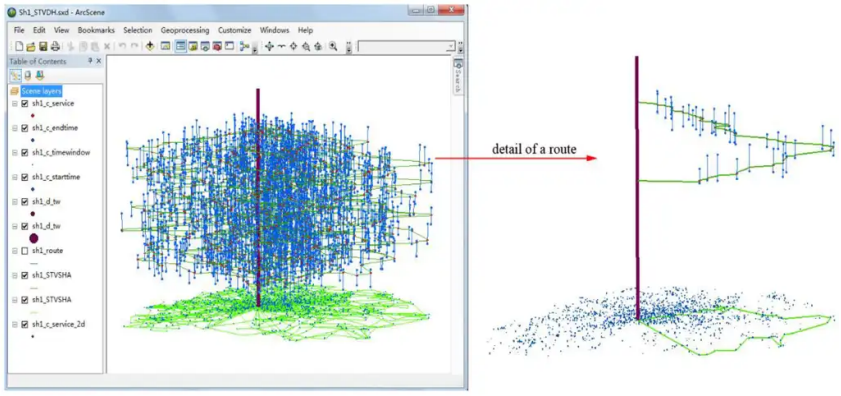
Table 4. Results from the large-scale VRPTW and MDVRPTW instances in Shanghai, China.

Instance	Type	N	M	D	Q	STVDH S_{avg} 10 (/km)	T_{avg} 10 (/min)	S_{Best} 10(/km)	T_{Best} 10 (/min)
Sh1a	VRPTW	2000	1	200	2000	59.3|3024.870	15.7	58|2996.104	15.9
Sh2a	VRPTW	4000	1	300	2000	119.8|4674.122	28.3	118|4598.592	30.6
Sh3a	VRPTW	6000	1	400	2000	171.0|6145.245	60.0	169|6052.084	65.9
Sh4a	VRPTW	8000	1	500	2000	220.3|7511.361	93.6	218|7372.345	98.9
Sh5a	VRPTW	10,000	1	600	2000	276.8|8829.127	120.8	274|8681.006	128.8
CNV						847.1		837	

Instance	Type	N	M	D	Q	STVDH S_{avg} 10 (/km)	T_{avg} 10 (/min)	S_{Best} 10(/km)	T_{Best} 10 (/min)
CTD(/km)						30,184.725		29,701.132	
Gap to S_{avg}								−1.60%	
Total Time (/min)							318.4		340.1
Sh1b	MDVRPTW	2000	2	200	2000	70.2|3045.601	16.1	69|2999.616	16.9
Sh2b	MDVRPTW	4000	3	300	2000	127.9|4265.365	29.8	126|4174.144	31.5
Sh3b	MDVRPTW	6000	4	400	2000	172.1|6247.152	62.2	169|5260.810	69.6
Sh4b	MDVRPTW	8000	5	500	2000	237.4|6849.848	98.4	235|6238.438	105.4
Sh5b	MDVRPTW	10,000	6	600	2000	298.8|6845.606	138.9	293|6784.751	139.5
CNV						906.4		892	
CTD (/km)						26,353.572		26,289.418	
Gap to S_{avg}								−1.73%	
Total Time (/min)							345.4		362.9

Figure 9. The result for large scale VRPTW problem Sh1a. (**a**) the final best solution; and (**b**) a spatial-temporal route.

Figure 9. The result for large scale VRPTW problem Sh1a. (**a**) the final best solution; and (**b**) a spatial-temporal route.



(**a**) (**b**)

compares the obtained results. It indicates that the STVDH presented in this study outperforms both ArcGIS and VRPEJ. In terms of efficiency, comparison with ArcGIS (total of 2141.1 min) and VRPEJ (total of 2355.3 min) indicates that the STVDH costs the least computing effort (total of 703 min). In terms of solution quality, the STVDH requires 1729 routes travelling 55,990.55 km to serve all customers in 10 instances. ArcGIS requires more vehicles (1764 routes) that travel a greater distance (62,047.809 km) to provide the same service. The VRPEJ, which utilizes a spatially kth nearest neighbors strategy, requires the most routes (1893 routes) that travelled 63,659.365 km. Hence, it confirms that the spatial-temporal Voronoi neighbor strategy in the presented STVDH is much better than the spatial neighbor strategy for local search to solve large scale VRPTWs.

Table 5. Comparison of the results of the STVDH with other heuristic algorithms.

Table 5. Comparison of the results of the STVDH with other heuristic algorithms.

Instance	Type	N	M	STVDH S_{Best} 10(/km)	STVDH T_{Best} 10 (/min)	ArcGIS S_{Best} 10(/km)	ArcGIS T_{Best} 10 (/min)	VRPEJ S_{Best} 10(/km)	VRPEJ T_{Best} (/min)

Instance	Type	N	M	STVDH S_{Best} 10(/km)	STVDH T_{Best} 10 (/min)	ArcGIS S_{Best} 10(/km)	ArcGIS T_{Best} 10 (/min)	VRPEJ S_{Best} 10(/km)	VRPEJ T_{Best} (/mi
Sh1a	VRPTW	1	2000	58|2996.104	15.9	62|3296.104	60.5	59|3285.404	50.8
Sh2a	VRPTW	1	4000	118|4598.592	30.6	120|4022.248	125.4	126|5234.168	122.
Sh3a	VRPTW	1	6000	169|6052.084	65.9	173|6738.745	197.2	190|6948.372	242.
Sh4a	VRPTW	1	8000	218|7372.345	98.9	223|8045.327	254.8	242|8438.982	290.
Sh5a	VRPTW	1	10,000	274|8681.006	128.8	278|9874.135	407.2	312|9814.247	428.
Sh1b	MDVRPTW	2	2000	69|2999.616	16.9	72|3308.245	60.5	74|3374.126	58.1
Sh2b	MDVRPTW	3	4000	126|4174.144	31.5	124|4610.283	125.4	132|4878.785	142.
Sh3b	MDVRPTW	4	6000	169|5260.810	69.6	174|6645.397	197.2	180|7013.522	248.
Sh4b	MDVRPTW	5	8000	235|6238.438	105.4	239|6982.136	254.8	260|6989.971	280.
Sh5b	MDVRPTW	6	10,000	293|6784.751	139.5	299|7525.189	568.1	339|7881.788	490.
CNV				1729		1764		1893	
CTD (/km)				55,990.550		62,047.809		63,659.365	
Gap to STVDH						10.82%		13.70%	
Total Computing Time (/min)					703.0		2141.1		2355

6. Discussion

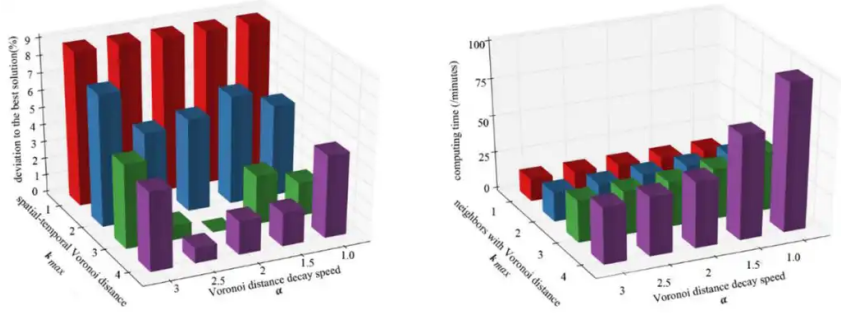
Taking the large scale VRPTW and MDVRPTW in Shanghai, China as an example, this section discusses the impact of the spatial-temporal Voronoi diagram and spatial-temporal proximity behind the best found results.

6.1. Impact of the Spatial-Temporal Voronoi Diagram

To evaluate the impact of the Voronoi diagram, this study investigated the balance of solution quality and computing time with different values for parameters k_{max} and α. As illustrates, considering the spatial-temporal Voronoi neighbors, an increase in k_{max} improves solution quality but decreases computing efficiency because more spatial-temporal Voronoi neighbors are involved in local search procedures. With regard to the effect of the Voronoi distance decay strategy, as a shows, both the slow decaying speed (α = 1) and the fast decaying speed (α = 3) generate worse results. However, as b shows, as a fast decaying speed requires less evaluation on far Voronoi neighbors, the STVDH consumes less computing efforts as the parameter α increases.

Figure 10. The impact of the spatial-temporal Voronoi diagram on the STVDH. (**a**) Deviation to the best solution; and (**b**) computing time.

Figure 10. The impact of the spatial-temporal Voronoi diagram on the STVDH. (**a**) Deviation to the best solution; and (**b**) computing time.



(**a**) (**b**)

6.2. Spatial-Temporal Proximity Analysis on the Best Found Results

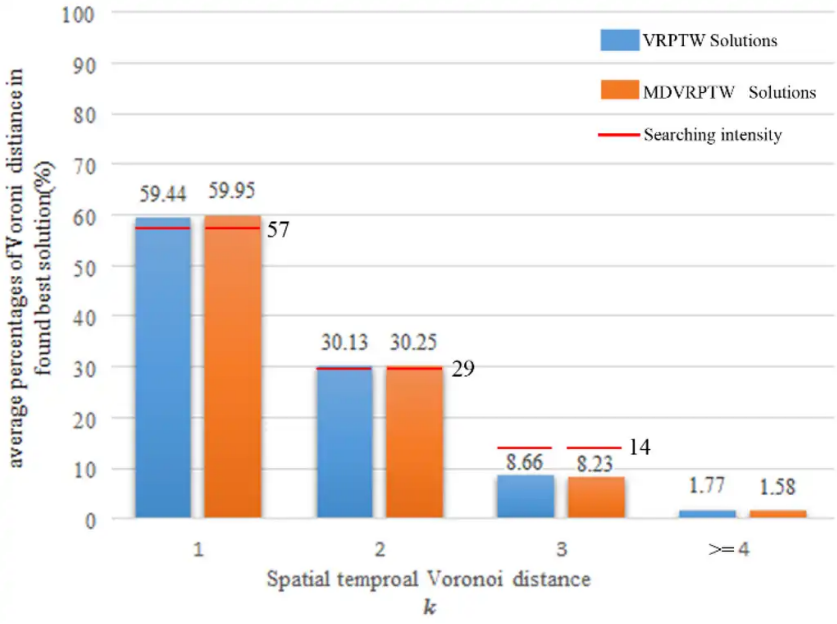
To understand spatial-temporal proximity behind the obtained results, we conducted an analysis on the Voronoi distance between consecutively-visited customers in the route of best found solutions in Section 5.2. displays the average percentage of Voronoi distance in the best found solutions. Two typical features are indicated below.

- The number of larger Voronoi distances is only a small proportion of the best found results. As displays, the percentages of Voronoi distances greater than three in the VRPTW and MDRPTW solutions are only 1.77% and 1.58%, respectively. Such a distribution agrees with the setting of parameter k_{max} in Section 5.2. Therefore, the spatial-temporal Voronoi neighborhood $k_{max}=3$ is very typical in the found solution.
- The percentage decreases sharply as the Voronoi distance increases. For the large-scale VRPTW dataset (Sh1a–Sh5a), the percentages for Voronoi distances 1, 2, 3, and >= 4 are 59.44%, 30.13%, 8.66%, and 1.77%, respectively, which is similar to the distribution in the MDVRPTW solution. This verifies that there is a spatial-temporal local compact structure in the routes of best found solutions. Compared with the theoretical searching intensity (as Equation (7) in Section 4.2.1 where) in the Voronoi distance decay strategy, these percentages systematically slightly shift to small Voronoi distances. This result demonstrates the effectiveness of the Voronoi distance-decay strategy, which searches more on near neighbors in the local search but still spends necessary efforts on far neighbors. $k_{max}=3$, $\alpha=2$

In summary, analysis of best found results demonstrates reasons why the spatial-temporal Voronoi diagram is effective in the STVDH algorithm.

Figure 11. Average percentage of the Voronoi distance in the best found solutions.

Figure 11. Average percentage of the Voronoi distance in the best found solutions.



7. Conclusions

Vehicle routing designs the least costly routes to satisfy the geographical distribution of human needs. Embedded in the GIS environment, it not only benefits transportation decision-making for both public and private sectors, but also enriches the value of geoinformation with the assistance of

location-based service. Inspired by spatial-temporal proximity, this article presents a novel spatial-temporal Voronoi diagram-based heuristic approach to solve large-scale VRPTWs quickly. The derived spatial-temporal Voronoi neighborhood measures proximity considering both spatial and temporal issues. The used Voronoi neighbors limits searching space in the construction algorithm, local search, and spatial-temporal feature-guided search. Moreover, the presented Voronoi distance-decay strategy assigns more searching efforts on spatial-temporal near neighbors.

Experiments on the VRPTW benchmarks and large-scale VRPTW instances in Shanghai, China with 2000–10,000 customers have demonstrated the good performance on VRPTW and MDVRPTW problems of different sizes. The obtained results indicate that the STVDH presented in this study can provide high quality solutions for large-scale VRPTWs in a reasonable time with the help of a spatial-temporal Voronoi diagram. It also verifies that the spatial-temporal proximity is very typical in best found solutions as 98% of segments had a Voronoi distance less than three.

The main contributions of this article are twofold. First, considering both time windows and spatial locations, the spatial-temporal Voronoi diagram is proposed to accelerate the solving process for VROs. In contrast to the spatial proximity based speeding up strategies [25,26,27], it makes use of spatial-temporal Voronoi neighbor to further reduce computing effort for local search based heuristics. The idea behind the presented approach that makes uses of spatial principles to accelerate complex optimizing processes can be extended to facilitate other space-related decision-making problems such as near real-time emergency response and large-scale facility location. Second, a spatial-temporal Voronoi diagram-based heuristic was developed to solve large-scale VRPTWs. This novel approach exhibits good performance on super large-scale VRPTWs up to 10,000 customers, which can cope with challenges from many complex real-world transportation and logistics applications.

Among future developments that we intend to undertake, we plan to integrate the presented approach with outdoor/indoor ubiquitous positioning and friendly navigating technologies for vehicles with cloud GIS. The presented effective approach will be used to upgrade the network analyst module in SDSSs to provide a more flexible transportation service for daily large-scale logistics and distribution activities. It will further enhance the spatial intelligence of modern transportation and logistics applications.
