International Journal of
Geo-Information

Article

A Parallel-Computing Approach for Vector Road-Network Matching Using GPU Architecture

Bo Wan [1,2], Lin Yang [1,3,*], Shunping Zhou [1,2], Run Wang [1,2], Dezhi Wang [1,2] and Wenjie Zhen [1]

1 Faculty of Information Engineering, China University of Geosciences (Wuhan), 388 Lumo Road, Wuhan 430074, China; magicwan1105@163.com (B.W.); zhoushunping@mapgis.com (S.Z.); runwang@cug.edu.cn (R.W.); wuhan1990hk@126.com (D.W.); zhenwenjiezwj@outlook.com (W.Z.)
2 National Engineering Research Center of Geographic Information System, Wuhan 430074, China
3 State Key Laboratory of Geo-Information Engineering, Xi'an 710054, China
* Correspondence: yanglin@mapgis.com; Tel: +86-159-2732-2600

Received: 2 October 2018; Accepted: 5 December 2018; Published: 7 December 2018

Abstract: The road-network matching method is an effective tool for map integration, fusion, and update. Due to the complexity of road networks in the real world, matching methods often contain a series of complicated processes to identify homonymous roads and deal with their intricate relationship. However, traditional road-network matching algorithms, which are mainly central processing unit (CPU)-based approaches, may have performance bottleneck problems when facing big data. We developed a particle-swarm optimization (PSO)-based parallel road-network matching method on graphics-processing unit (GPU). Based on the characteristics of the two main stages (similarity computation and matching-relationship identification), data-partition and task-partition strategies were utilized, respectively, to fully use GPU threads. Experiments were conducted on datasets with 14 different scales. Results indicate that the parallel PSO-based matching algorithm (PSOM) could correctly identify most matching relationships with an average accuracy of 84.44%, which was at the same level as the accuracy of a benchmark—the probability-relaxation-matching (PRM) method. The PSOM approach significantly reduced the road-network matching time in dealing with large amounts of data in comparison with the PRM method. This paper provides a common parallel algorithm framework for road-network matching algorithms and contributes to integration and update of large-scale road-networks.

Keywords: parallel computing; road-network matching; GPU architecture; PSO

1. Introduction

1.1. Road-Network Matching

Road-network matching is a key technology of vector road-map integration, fusion, and update [1]. The main task of road-network matching is building the corresponding relationship of road–object pairs that represent the same segment of real-world road in heterogeneous road maps [2]. It has significant potential for the timely and cost-effective updating of road-network data and geographic-information science applications (e.g., vehicle-navigation products) [3].

The core of the road-network matching method is evaluating the similarity of two node/road features and determining the corresponding relationship of matching pairs.

(1) Similarity quantifies the similar degree of two features, providing the basis for determining the matching relationship of homonymous objects [1]. The similarity-evaluation function varies with the matching unit. In earlier studies, the matching unit is usually the limited local context around

nodes or road segments. Such similarities as the Hausdorff distance or the Fréchet distance of homonymous nodes or road segments and the topological structure of intersections are mainly considered in these approaches [4–8]. Some researchers also integrated multi-factors (e.g., length, orientation, and number of topological connections) for road-segment matching [2,9]. However, matching a unit at the node/road segment scale may bring about local optimization problems, limiting matching accuracy. Some studies pay more attention to the structural information of the road network and shift the matching unit to a larger scale, such as junction/segment clusters, for global optimization [3,10]. It makes the similarity measures more comprehensive but more complicated.

(2) When the similarity of matching units is evaluated, the matching relationship of the nodes/road segments can be determined. Most earlier methods rely on a sequential greedy strategy [1]. This strategy intends to achieve the local optimum by comparing similarity with an experiential threshold [4,11,12]. To overcome the limitation of the local optimum, several optimization strategies have been proposed to find a global optimal solution from all possible matching choices [8,13]. In these strategies, the objective function maximizes total similarity among all matched feature pairs. Global optimal solutions have greatly promoted matching accuracy. However, increased computational complexity also degrades the performance of the matching method.

To date, most studies on road-network matching focus on evaluating the effectiveness of the matching model and promoting matching accuracy. Few studies have mentioned the performance of the matching method, and are listed in Table 1. The amount of data in most studies is small, fewer than 1000 nodes/segments. Time cost varies from seconds to hours. Even at the same data scale, the time cost is different. Reasons that affect the computational costs can be summarized in three points.

Table 1. Studies that mention the performance of road-network matching algorithms (listing data scale, time cost, accuracy, recall, and environment).

Author (Year)	**Data Scale**		**Time Cost**	**Accuracy Ratio**	**Recall Ratio**	**Environment**
	Node Number	**Segment Number**				
Li (2010) [1]	/	434/423	31 min 39 s	98%	-	-
	-	308/264	23 min 16 s	97.58%		
	-	377/374	1 min 37 s	97.85%		
	-	344/322	2 h 6 min 14 s	95.03%		
Tong (2014) [8]	-	99/84	1.8 s	91.8%	100%	MATLAB 6.0, Intel Core 5 Duo processor, and 4 GB memory.
	-	116/112	3.83 s	96.1%	96.1%	
	-	137/108	4.71 s	87.2%	91.2%	
Zhang (2012) [15]	249/387 (Wuhan)	358/577 (Wuhan)	4 s per iteration	97.2%	91.2%	Microsoft Visual Studio 2008 (C# Programming Language) and ArcGIS Engine 9.3.
	1059/1299 (Beijing)	1567/2019 (Beijing)	5 s per iteration	96.5%	94.6%	
	2732/1666 (Zurich)	3995/2489(Zurich)	30 s per iteration	96.5%	94.7%	
Luan (2012) [16]	700	-	13.5 min	86.11%	90.30%	CPU 3.06 GHz and 1 GB of RAM.
Yang (2014) [3]	756/361	-	-	95.45%	98.91%	-
Liu (2016) [17]	-	arcs: 577/585; strokes: 136/140 (Ring and radial area)	25.44 s	80.67%	-	-
	-	arcs: 522/513; strokes: 113/114 (Grid area)	10.43 s	76.83%		
	-	arcs: 1092/1101; strokes: 220/230 (Hybrid area)	43.56 s	83.33%		
Zhao (2010) [14]	121/261	142/325	71 s	-	-	C# Programming Language and ArcGIS Engine 9.2
Fan (2016) [18]	-	12208/5945 (Heidelberg)	22 min	98.3%	95.3%	-
	-	107438/61559 (Shanghai)	142 min	96.9%	85.6%	
Zhang (2018) [19]	474/671 (Beijing)	784/962 (Beijing)	44 s	95.3%	95.0%	C #, ArcEngine10.1, CPU Intel Core 2 Duo E7200, 3.53 GHz.
	1556/2229 (Shanghai)	3250/4480 (Shanghai)	247 s	95.1%	95.9%	

(1) The scale of road-network data. Dealing with a larger scale of road-network data requires a higher time cost, because the number of objects that need to be computed increases.
(2) The complexity of the matching unit and the similarity function. Compared with a simple matching unit (e.g., road segment or node), the similarity function of complex matching units (e.g., node/segment cluster) is always more complex in order to take more factors into careful consideration. Thus, complex matching units often need more computing time.
(3) The number of iterations and convergence rate. Optimal solutions are usually determined iteratively in the previous literature. For instance, the probabilistic relaxation method iterates to find the stable matching probability matrix, and its computing time is closely related to convergence rate and the number of iterations [14].

From a data perspective, dividing the data into smaller units can offer a potential for accelerating matching efficiency by parallel computation [1].

1.2. Research Object

Numerous algorithms have been proposed to solve the road-network matching problem [1–5,8,13,14,19]. The accuracy of road-networks matching is greatly promoted in previous studies. However, the performance of the matching method is less noticed. As the sizes of the databases increase, the performance issue becomes increasingly prominent. Parallel techniques are well-suited to accelerate matching computation. Dividing large areas into several sub-regions could be an effective way for faster computing with parallel computation [1]. Graphics-processing units (GPU) that have hundreds, even thousands, of integrated cores in a single chip are perfect for solving problems that can be partitioned into independent and smaller parts.

Accordingly, to perform the road-network matching in an effective and efficient manner, a parallel matching strategy using GPU architecture is developed in this paper, and particle-swarm optimization (PSO) is introduced for the global optimization of the matching process. PSO is straightforwardly parallelizable and has shown excellent convergence rates, which can further reduce the computing time. It has the advantage of concurrently evaluating a model population, and particles can be evenly distributed to multiple threads/processes [20]. The remainder of this article is organized as follows. Section 2 introduces the model construction of the particle-swarm optimization-based matching (PSOM), including the matching unit introduction, the mapping of matching process to particle flight process and the workflow of PSOM. Section 3 introduces the parallel strategies of similarity computation and matching identification in PSOM. Section 4 describes the accuracy evaluation indices. Experiments conducted on 14 datasets with different scales are presented in Section 5.1. Parameter settings of particle and thread are discussed in Section 5.2; algorithm accuracies are analyzed in Section 5.3; and Section 5.4 examines the speed-up ratio of PSOM under the GPU environment, comparing with another global optimization strategy, the probability-relaxation algorithm (PRM). The conclusions are drawn in Section 6.

2. PSO-Based Road-Network Matching Algorithm (PSOM)

2.1. Matching Unit

The node/road-segment-matching unit may result in local optimization problems. In this paper, the “stroke”, a natural morphological extension of continuous road segments, is employed as the matching unit. Strokes can serve as reference to help locate homonymous road segments and handle a fuzzy matching relationship of road segments. A hierarchical stroke structure is also a global constraint that can avoid local matching errors caused by nonsystematic bias.

2.2. Model Construction of PSOM

PSO has the advantage of global optimization and parallelization. This section provides a detailed solution of how to use the PSO to solve road-network matching problems.

(1) Particle Swarm

The first step of PSO is to generate a swarm composed of several models (position vector) in the model parameter space. The initial models can be defined by a given random distribution. Each model is represented by a particle that interacts with its neighborhood to find the global minimum of the misfit function.

The swarm is composed of m particles, which means m potential-candidate solutions in the solution space. The swarm in the nth iteration is denoted as $P(n)$ = $[P_1(n),P_2(n), \ldots ,P_m(n)]$, where $P_m(n)$ is the mth particle in the nth iteration.

Suppose the dimension of one particle is d, the ith particle can be represented as position vector $P_i(n)$ = $[p_{i1}(n),p_{i2}(n), \ldots ,p_{id}(n)]$, where p_{ij} (n)denotes the position of j-dimension for particle i.

Road-network matching is an optimization problem in d (the number of nodes in road network) dimensional space. In this paper, one particle corresponds to one solution space/model for the road-network matching. A particle's position in d-dimensional space is a solution composed of d matching pairs.

We denote the road network with fewer nodes as R_a, and the road network with more nodes as R_b. The nodes of R_a are labeled as N_a, and the nodes of R_b are labeled as N_b. The size of N_a is viewed as the dimension d of the model, and each node in N_a corresponds to one dimension of the model. Actually, the size of N_a is used to define the size of position vector $P_i(n)$ that is denoted as dimension d. Then, the position of the ith dimension for particle i, p_{ij} (n) is represented by a matching pair (n_{aj}, n_{bx}), $n_{aj} \in N_a$, $n_{bx} \in N_{pb} \in N_b$. n_{aj} is the jth element in N_a and n_{bx} is any node from the potential candidate nodes set $N_{pb,}$ which also belongs to $N_{b.}$

Specifically, suppose feature n_{aj} in R_a has k potential candidate features $\{n_{c1},n_{c2}, \ldots , n_{ck}\}$ in R_b, then all potential matching pairs for n_{aj} constitute set $PSet_{aj}\{(n_{aj}, n_{c1}), (n_{aj}, n_{c2}), \ldots , (n_{aj}, n_{ck})\}$. Each (n_{aj}, n_{bx}) denotes one matching pair of R_a and R_b, and represents the position for one specific dimension. Thus, the set of (n_{aj}, n_{bx}), which is composed of m matching pairs, constitutes a specific solution space, and corresponds to a specific position of the particle in the M-dimensional space. The matching pair (n_{aj}, n_{bx}) updates during the flight process in terms of iterative function; therefore, the position of the particle and the model vector change.

(2) Particle Velocity

Particle velocity controls how a particle moves in the model parameter space. The velocity vector represents the changing degree of the particle position. Velocity vector is adjusted according to its own personal best model and the global best model of the whole swarm. It is represented by $v_i(n)$ = $[v_{i1}(n),v_{i2}(n), \ldots v_{id}(n)]$, in which $v_{ij}(n)$ is the velocity of particle i in the jth dimension on the nth iteration. Here, $v_{ij}(n)$ represents the updated extent of the jth matching pair, which will decide the next matching pair for $n_{aj.}$

(3) Particle Fitness Value

Particle fitness value is determined by the optimization objective, which is used to evaluate the optimization extent. The particle fitness value is the final optimized solution when the iteration process ends. In our problem, the optimization objective is the global similarity of R_a and $R_{b.}$ Here, the fitness of one particle is represented by the sum of the similarity of each matching pair in a particle. The greater the fitness value is, the better the matching results are.

$$PFV_i = \sum_{i=0}^{d} Sim(n_{aj}, n_{bx}) \tag{1}$$

The fitness value of particle i in the nth iteration is represented as $f_i(n) = [f_{i1}(n), f_{i2}(n), \ldots f_{id}(n)]$, in which $f_{ij}(n)$ is the fitness value for the jth dimension. The similarity-measure model Sim(n_{aj}, n_{bx}) is used to calculate $f_{ij}(n)$ for jth matching pair (n_{aj}, n_{bx}) in the particle (see Appendix A). Note that here $f_{ij}(n)$ is a universal framework that could fit different kinds of similarity functions.

$$f_{ij}(n) = \mathrm{Sim}(n_{aj}, n_{bx}) \tag{2}$$

Sim(n_{aj}, n_{bx}) is the similarity between node n_{aj} and n_{bx}. $f_{ij}(n)$ quantitatively describes the similarity between node n_{aj} and n_{bx}.

(4) Individual Optimum Extremum

Individual optimum extreme refers to the optimal solution in its multiple iterations for a single particle, which is also called the personal best model. The formula is as follows:

$$P_i^{cbest}(n) = \mathrm{Max}(PFV_i(n)), n \in [1, ci] \tag{3}$$

where $P_i^{cbest}(n)$ is the maximum PFV chosen from the initial state to the current iteration ci for particle i. $P_i^{cbest}(n)$ is also represented as vector $P_i^{cbest}(n) = [P_{i1}^{cbest}(n), P_{i2}^{cbest}(n), \ldots P_{id}^{cbest}(n)]$, where $P_{ij}^{cbest}(n)$ is the value on the jth dimension. In this way, the local optimization of each matching pair is also achieved by updating the personal best model of each particle. Thus, the individual optimum extreme refers to a set of matching pairs with individual best fitness.

(5) Global Optimal Extremum

Global extreme value refers to the optimal particle in the process of searching from the initial state to the current iteration for the whole particle swarm, which is also called the global best model. In each iteration, the particle with maximum fitness PFV is selected from the particle swarm. $P^{gbest}(n)$ is continuously updated in the iteration process.

$$P^{gbest}(n) = \mathrm{Max}(PFV_i(n)), i \in [1, m], n \in [1, ci] \tag{4}$$

The global optimal extremum on the nth iteration is represented as $P^{gbest}(n) = [P_1^{gbest}(n), P_2^{gbest}(n), \ldots P_j^{gbest}(n)]$, where $P_j^{gbest}(n)$ is the value on the jth dimension. In this way, the flying process maximizes the total similarity of all matched feature pairs of R_a and R_b. When the iteration ends, $P^{gbest}(n)$ is the optimal matching result for R_a and R_b.

(6) Particle Flying

The principle of particle flying is illustrated in Figure 1; the velocity of particle i, $v_{ij}(n+1)$ is decided by three parts: (1) velocity in the last iteration, which inherits the previous velocity; (2) distance between the current position of particle i and the optimal position of particle i, which represents the cumulative experience of particle i; and (3) distance between the current position of particle i and the global optimal particle in the swarm, which represents the sharing experience from the society. In this way, other particles gradually fly to the state of global optimum.

The velocity and position of each particle are updated as follows:

$$v_{ij}(n+1) = wv_{ij}(n) + c_1 r_1 (P_{ij}^{cbest}(n) - x_{ij}(n)) + c_2 r_2 (P_j^{gbest}(n) - x_{ij}(n)) \tag{5}$$

where $v_{ij}(n+1)$ means the velocity of particle i on the jth dimension at iteration $n+1$. w is an inertia weight, and c_1 and c_2 are two acceleration parameters that, respectively, control the cognition and social interactions of the particles. r_1 and r_2 are uniform random-number vectors drawn at iteration n, $P_{ij}^{cbest}(n)$ is the personal best model of particle i on the jth dimension at iteration n, and $P_j^{gbest}(n)$ is the global best model of the swarm on the *j*th dimension at iteration n. Through particle iteration, the optimal similarity matching-pair sequence is gradually formed.

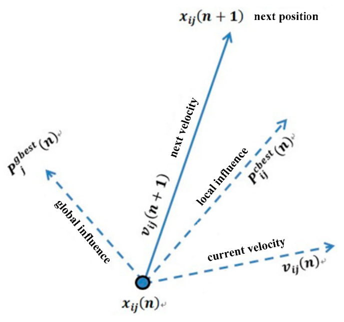


Figure 1. Principle of particle flying.

Then, the next position of each particle is calculated by Formula (6):

$$x_{ij}(n+1) = x_{ij}(n) + v_{ij}(n+1) \tag{6}$$

To identify the matching pair, we consider all the features in R_a as the anchor point; in other words, n_{aj} in the *j*th matching pair (n_{aj}, n_{bx}) is fixed. The objective is to find the correct matching feature from N_{pb} for each feature n_{aj} in R_a. Since each feature n_{aj} in R_a has the potential matching area in R_b, in our problem, the flying range of each dimension for one particle is limited. The model parameter space of the *j*th dimension is restricted to all potential matching pairs of $n_{aj,}$ which is selected from set $PSet_{aj}$. The flying position range for *j*th dimension should be restricted in $PSet_{aj.}$

Usually, the change of velocity is a continuous process, while in the search of matching pairs, the change of the matching pair is a discrete process, so we need to map the continuous changing process into a discrete process. Meanwhile, to guarantee that the changing direction of velocity in each iteration process is in the right orientation, the similarity value is utilized in the personal best model of particle *i* and the global best model of the swarm to ensure that global similarity is increasing in the flying process.

The speed-range vector is defined to restrict the flying extreme. The *j*th dimension of this vector represents the flying range for the *j*th matching pair, $Vr_j \in$ [minSim($PSet_{aj}$)-maxSim($PSet_{aj}$), maxSim($PSet_{aj}$)-minSim($PSet_{aj}$)]. The lower bound is defined by minSim($PSet_{aj}$)-maxSim($PSet_{aj}$), and the upper bound is defined by maxSim($PSet_{aj}$)-minSim($PSet_{aj}$), in which minSim($PSet_{aj}$) is the minimum value among the similarity value for k potential candidate pairs in $PSet_{aj}$, and maxSim($PSet_{aj}$) is the maximum value among the similarity value for k potential candidate pairs in $PSet_{aj}$.

For each dimension, k potential-candidate pairs in $PSet_{aj}$ are sorted in order of similarity value, and the corresponding relationship between the similarity value interval and matching pairs should be established for these k candidate pairs. In the iteration process, to get a new position for the *j*th dimension of particle *i*, a matching pair (n_{aj}, n_{bx}) is found that has the closest value with x_{ij} from $PSet_{aj.}$. Then, Sim(n_{aj}, n_{bx}) is assigned to x_{ij}.

Mapping between continuous velocity and discrete matching is achieved. Using the extent of similarity changing to update speed is more in line with the optimization objective of global matching.

Fitness value *PFV* of each particle changes synchronously when all matching pairs in the particle are changed in terms of iteration function. That is, the position of the entire particle swarm changes with the change of velocity position of each dimension. In this way, the optimization direction of the particle swarm is in the direction of global similarity increasing in the process of iterations.

2.3. PSOM Workflow

The PSOM algorithm can quickly find optimal solutions, but it is easy to fall into a local optimum. Therefore, a tabu-search strategy is used to prevent prematurely falling into a local optimum.

Two optimal solutions should be recorded in the iterations. $P^{gbest}(n)$ records the global best model of the swarm at iteration n, and $P^{hgbest}(n)$ records the history global best model of the swarm in the process of n iterations. $P^{gbest}(n)$ is put into the tabu list to stay for $\sqrt{N}$ time, in which N is the number of particles. The particle in the tabu list will not be selected except for when it meets the contempt criterion that intends to prevent loss of the global optimal solution. The contempt criterion is valid when the PFV of the current particle is greater than $P^{hgbest}(n)$.

The PSOM workflow is shown in Figure 2, and the specific steps are as follows:

(1) Similarity calculation. For all nodes in R_a, calculate the similarity value of each matching pair in $PSet_{aj}$ using Equation (2), and sort $PSet_{aj}$ by the similarity value.
(2) Particle Initialization. Initialize particles in a random manner, and select $P^{gbest}(n)$, putting into the tabu list.
(3) Particle Iteration. Calculate and update the velocity and position of each particle using Equations (5) and (6). Calculate the PFV for all particles at iteration n using Equation (1), and the maximum particle is selected to be $P^{gbest}(n)$. If the PFV is greater than $P^{hgbest}(n)$, the maximum particle is assigned to $P^{hgbest}(n)$; otherwise, $P^{gbest}(n)$ is put into the tabu list if it is not contained. The global optimal model $P^{gbest}(n)$ and the personal best model $P_i^{cbest}(n)$ are updated and recorded in each iteration. Continue with this step until the maximum iteration number is reached.
(4) Output all the matching pairs in the final global optimum particle $P^{hgbest}(n)$.

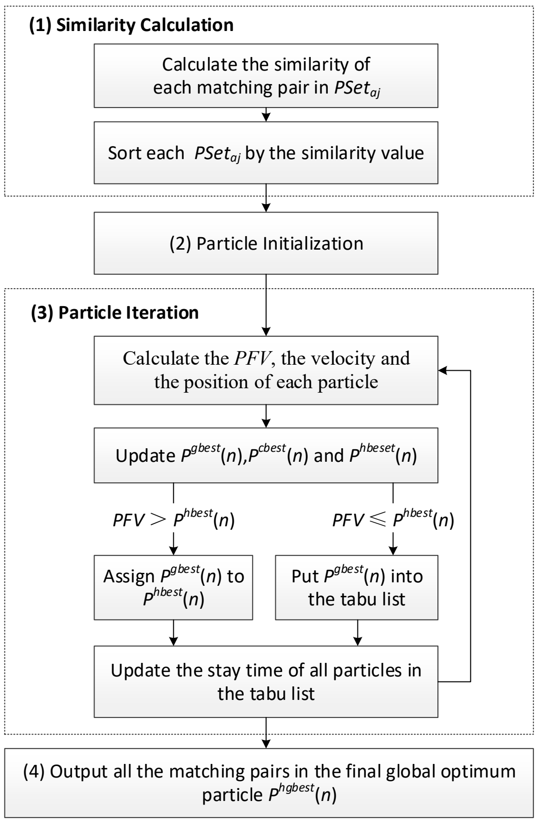

(1) Similarity Calculation
Calculate the similarity of each matching pair in $PSet_{aj}$
Sort each $PSet_{aj}$ by the similarity value
(2) Particle Initialization
(3) Particle Iteration
Calculate the PFV, the velocity and the position of each particle
Update $P^{gbest}(n)$,$P^{cbest}(n)$ and $P^{hbeset}(n)$
$PFV > P^{hbest}(n)$
$PFV \leqslant P^{hbest}(n)$
Assign $P^{gbest}(n)$ to $P^{hbest}(n)$
Put $P^{gbest}(n)$ into the tabu list
Update the stay time of all particles in the tabu list
(4) Output all the matching pairs in the final global optimum particle $P^{hgbest}(n)$

Figure 2. Particle-swarm optimization-based matching (PSOM) workflow.

3. PSOM Parallelization Strategy

PSOM is composed of a similarity computation process and matching-relationship identification process. Accordingly, the parallelization strategy is designed for these two stages, respectively (Figure 3).

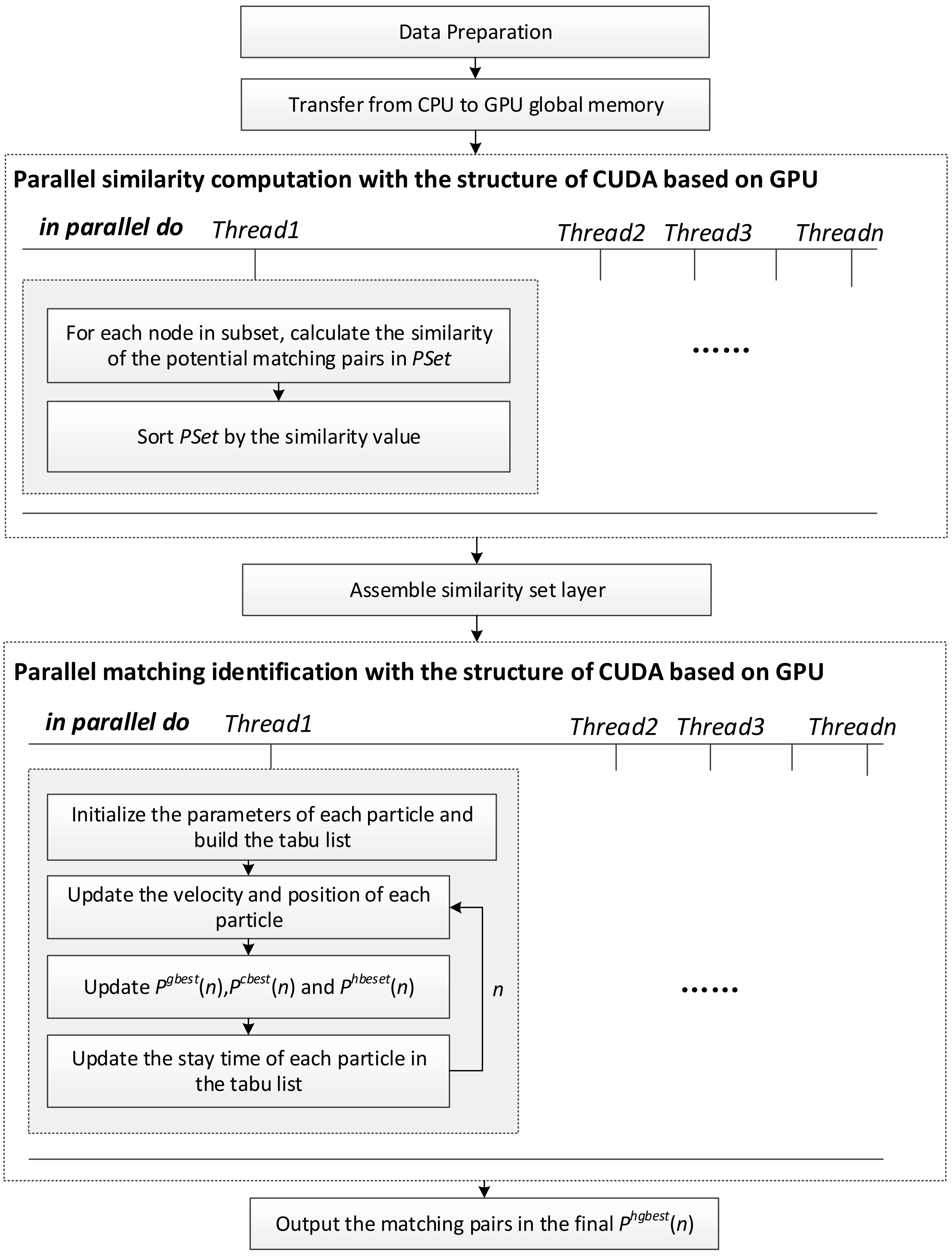

Data Preparation
Transfer from CPU to GPU global memory
Parallel similarity computation with the structure of CUDA based on GPU
in parallel do
Thread1
Thread2
Thread3
Threadn
For each node in subset, calculate the similarity of the potential matching pairs in PSet
Sort PSet by the similarity value
……
Assemble similarity set layer
Parallel matching identification with the structure of CUDA based on GPU
in parallel do
Thread1
Thread2
Thread3
Threadn
Initialize the parameters of each particle and build the tabu list
Update the velocity and position of each particle
Update P^gbest(n), P^cbest(n) and P^hbeset(n)
n
Update the stay time of each particle in the tabu list
……
Output the matching pairs in the final P^hgbest(n)

Figure 3. Particle-swarm optimization-based matching (PSOM) parallelization strategy.

3.1. Parallel-Similarity Computation

Similarity calculation is the precondition of matching-relationship identification. The time cost of similarity calculation is dependent on the data scale and the complexity of the similarity measure. Transforming similarity calculation from a sequence way to a parallel way can significantly speed up the process, and is applicable for different similarity-measurement methods.

For each node n_{aj} in the original map, the potential matching pairs are selected using a specified buffer threshold. Suppose each n_{aj} has m matching candidates, stored in $PSet_{aj}\{(n_{aj}, n_{c1}), (n_{aj}, n_{c2}), \ldots, (n_{aj}, n_{cm})\}$. The similarity for n nodes in the original map need $n \times m$ similarity operations. Similarity computation of matching pairs is an independent operation, so the *n*m* similarity operations can be distributed across multiple threads, and the whole similarity-calculation process can be parallelized.

There are three phases for the parallelization of similarity computing. The first phase is decomposing node–feature pairs into multiple subsets for threads. The size of subsets is determined by the total number of nodes in the original map and the number of available compute units. The second phase is a parallel execution of similarity operations. Each thread handles a subset, calculating the similarity of node–feature pairs and sorting *PSet* by the similarity value. The final phase is collecting the outputs derived from each thread and recompiling them into a large single similarity set, which is used for the matching-relationship identification process.

Additionally, due to the diversity of data structures in different data sources, the original data are reconstructed in the data preparation stage. The additional structure (i.e., stroke structure) and the matching candidates of each node are all prepared beforehand. They are allocated in the global memory for the usage of similarity calculation in the GPU procedure.

3.2. Parallel-Matching Identification

The flight behavior of each particle is independent and has intrinsic parallel attributes, which make the PSOM algorithm a kind of natural parallel optimization algorithm.

The matching-relationship identification of PSOM utilizes a decomposition parallel strategy. This strategy decomposes the whole particle swarm into subgroups corresponding with the compute units. Each node of the compute units deals with all the tasks including initialization, iterative information update, and information exchange. The best-location information is exchanged between nodes according to a specific strategy. Because PSOM is a fine-granularity parallel-computing model, the node number (i.e., the thread number of GPU) is set equal to the particle number in this paper. This can balance the computing load and maximize parallel the efficiency.

The PSOM algorithm can be divided into two stages, particle initialization and particle flight. Two GPU kernel functions are designed to parallel these two processes, respectively. The first kernel function ***k-particle-initial*** initializes the particles in parallel. The number of threads is determined according to the number of particles, and the *curand* function is used to generate random numbers that are used to set the initial position of particles in parallel. The initial potential matching pairs in one particle are randomly selected from the $PSet_{aj}$ of nodes in R_a. Particles with different possible parameter models are initialized in parallel. After the initialization of all particles is completed, the comparison is made in the first thread to find the global optimal value and local optimal value. The second kernel function ***k-particle-fly*** is iteratively executed in parallel. The iteration number of ***k-particle-fly*** is a preset flying-time number, and the number of threads is still determined by the number of particles. The flying process is described in detailed in the pseudo-code (Algorithm 1).

In the flying process, communications are inevitable for visiting and updating the global optimum particle and the tabu list. The key point is to control synchronization access to the global memory. Thus, a shared region should be designed for the global optimum particle and the tabu list. The global memory is utilized to reserve the global optimum value. After the local optimum value in each particle is updated, the global optimum particle is calculated through communication between each single thread and the global memory. The global optimum particle is finally updated into the global memory.

To avoid the conflict of the global optimum updating and reducing communication times/cost, a periodically updated strategy is adopted. The update behavior is executed in the first thread when the specified iteration of all particles is completed. Hence, the global optimal value can be periodically updated to guide the following flying direction of each particle. This strategy not only fully exerts the independence of each particle, but also reduces the communication cost.

Besides, during iteration, many random values are needed to avoid frequent data exchange between CPU and GPU. The CUDA CURAND library on the GPU can be used to directly generate random numbers instead of transform time from CPU to GPU.

Algorithm 1: Parallel-Matching Relationship Identification

```
Input:
Road-network data in a defined GPU data structure; similarity set layer.
Output:
Matching relationship between road-network $R_a$ and $R_b$.
Steps:
1: Execute kernel function k-particle-initial.
        1.1: Number of threads is determined according to number of particles.
        1.2: The curand function is used to randomly set the initial position of each particle.
        1.3: Global optimal value and local optimal value are found in the first thread.
2: While iteration condition not reached do
    kernel function k-particle-fly.
        2.1: The velocity of each particle is calculated according to Equation (5)
        2.2: The position of each particle is updated according to Equation (6)
        2.3: The PFV of each particle is calculated.
        2.4: When all particles are done, the $P^{cbest}(n)$ and $P^{hbest}(n)$ of each particle and $P^{gbest}(n)$ of the whole particle swarm are updated.
        2.5: If PFV of the current particle is greater than $P^{hbest}(n)$, then the $P^{gbest}(n)$ is assigned to $P^{hbest}(n)$. Otherwise, $P^{gbest}(n)$ is put into the tabu list.
        2.6: The stay time of all particles in the tabu list is updated.
3. If the iteration condition is reached then
    the best particle found in the global memory is transferred from the GPU to the CPU.
4. Return the matching pairs in the final global optimum particle $P^{hgbest}(n)$, the algorithm ends.
```

4. Model Evaluation Indices

Two evaluation indices were used to access the accuracy of the models: the matching accuracy (Equation (7)) and the accuracy growth rate (Equation (8)).

The matching accuracy (P) is determined by the right matching features (TP), the wrong matching features (FP), and the ambiguous features that cannot be matched artificially (AM).

$$P = \frac{TP}{TP + FP + AM} \times 100\% \tag{7}$$

The closer that P value is to 1, the more correct matching features the algorithm recognizes.

The accuracy growth rate (AGR) is used for comparing the result accuracies between different algorithms.

$$AGR_{(A,B)} = \frac{P_A - P_B}{P_B} \tag{8}$$

where $AGB_{(A,B)}$ represents the accuracy growth rate of Algorithm A compared to Algorithm B. P_A is the matching accuracy of Algorithm A, and P_B is the matching accuracy of Algorithm B. A higher absolute value of AGR indicates a higher accuracy change. If AGR is greater than zero, Algorithm A performs more accurately than Algorithm B; otherwise, Algorithm B is better.

The effectiveness of performance optimization was evaluated by comparing with a benchmark. To the best of our knowledge, the road-network matching study using parallel computing strategy has not been reported in previous articles. Therefore, it is not possible to compare our parallel method with a similar parallelized method. In recent years, the probability-relaxation-matching algorithm (PRM) is often employed in road-network matching studies [5,13–15,19,21,22], thus it was adopted as the benchmark. The general probability-relaxation framework was utilized to implement the PRM in our experiment [23]. In addition, the similarity-computation function used in PRM is totally the same with that used in PSOM. The ending condition for the iteration of the PRM was set based on the experience value (0.0005) [15].

5. Results and Discussion

5.1. Experimental Environment and Data

To evaluating the performance of our method, 14 pairs of road networks were utilized with the node scale ranging from 44 to 72,130 (see Table 2 and Appendix B). Areas 1–6, 8–9, and 12 are from the Wuhan, China; Area 7 is part of the Chinese highway network; Areas 10–11 belong to Foshan, China; and Areas 13–14 belong to Oakland, New Zealand. Each road-network pair comes from different data producers. The source map represents the road network to be matched, and the destination map represents the road network that is used to match the source map.

Table 2. Fourteen pairs of road networks used for experiments. NS represents the node number of the source map, and ND represents the node number of the destination map.

No.	NS	ND	No.	NS	ND
Area 1	44	582	**Area 8**	519	673
Area 2	68	673	**Area 9**	582	825
Area 3	82	396	**Area 10**	4938	5530
Area 4	110	494	**Area 11**	7738	9049
Area 5	176	540	**Area 12**	10,820	13,961
Area 6	382	396	**Area 13**	13,815	16,771
Area 7	478	493	**Area 14**	72,130	127,205

The proposed parallel strategy of PSOM was implemented using C++ programming language with the structure of CUDA based on GPU (PSOM-GPU). Experiments were carried on a computer equipped with an NVIDIA GeForce GTX 960 GPU and an Inter(R) Pentium(R) D 3.00 GHz with 3.5 GB RAM. The software-development environment is composed of CUDA 8.0 SDK, Visual Studio 2010, and MapGIS 10 (ZONDY, Wuhan, Hubei, China).

5.2. Parameter Setting

Two parameters needed to be set in the experiments: the particle number and the thread number.

The particle number used in PSO algorithms is not uniform in different studies, and is usually between 20 and 50 in sequential algorithms and between 32 and 2000 in parallel algorithms [24–26]. To obtain an appropriate particle number, a set of particle numbers (i.e., 50, 100, 200, 300, 400, 1000, and 2000) were tested in ten experimental areas with an iteration number of 200.

The algorithm accuracies based on each different particle number were compared with the accuracy of particle number 50 which served as the baseline. As shown in Figure 4, AGR values change slightly when particle number is in the range of 100–300. However, as the particle number increases, most AGR values have a dramatic increase when the particle number is 400. After that, AGR curves remain relatively stable. Overall, the increase of particle number can promote the matching accuracy; however, more memory space and processing time are required. Take Area 7 (439 nodes) as an example, both the matching-relationship identification time and the total time cost increase as the particle number grows, and an especially dramatic increase appears when the particle number is larger than 400 (Figure 5). Thus, particle number is positively associated with accuracy, but negatively correlated with efficiency. Based on the above analysis, the particle number was set as 400 in this study, which requires less time while maintaining a good performance.

The thread number differed in the two parallel stages.

In the stage of similarity computation, more than one node–feature pair is assigned in one thread. Thus, different combinations of node–feature numbers (*FN*) (i.e., the number of node–feature pairs assigned to each single thread) and thread numbers (*TN*) were tested. *FN* was set from 5 to 25 with an interval of 5. The size of *TN* grows in multiplicative size, from 16 to 256. Four road-network sets (Area 10, Area 11, Area 12, and Area 14) were employed to illustrate the time consumption of similarity computation under different conditions (Figure 6). For Areas 10–12, the time cost increased with the growth of *FN* and *TN*. No matter what value *FN* takes, the similarity computation time cost with 16 threads was always the least among five different settings of thread numbers. Thus, for the first three sets, the lower *FN* and *TN* are set, the faster is the computation speed. However, this regularity does not exist for Area 14. The experimental result reveals that, when setting parameters of *FN* = 15 and *TN* = 32, the time cost is the least, which is the best solution for similarity computation process in this study.

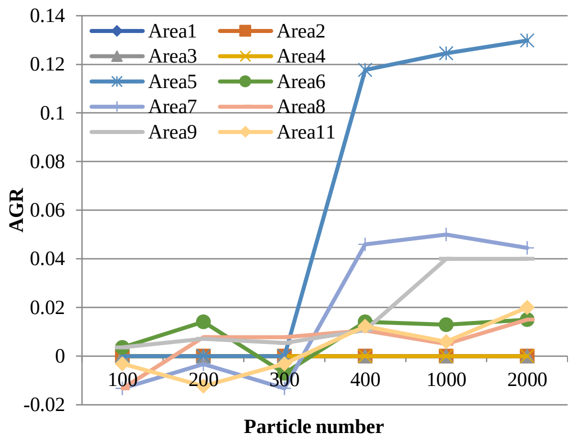


Figure 4. Relationship between accuracy growth rate and particle number.

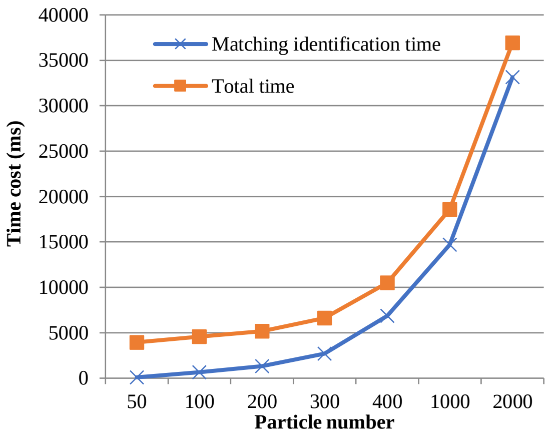


Figure 5. Matching identification time and total time cost under different particle numbers using Area 7 (439 nodes) as an example.

In the matching-relationship identification stage, the number of threads is equal to the total number of particles. Each thread performs the same operation simultaneously and synchronously. Since the particle number was set to 400, the thread number of matching-relationship identification was also set to 400 in our experiment.

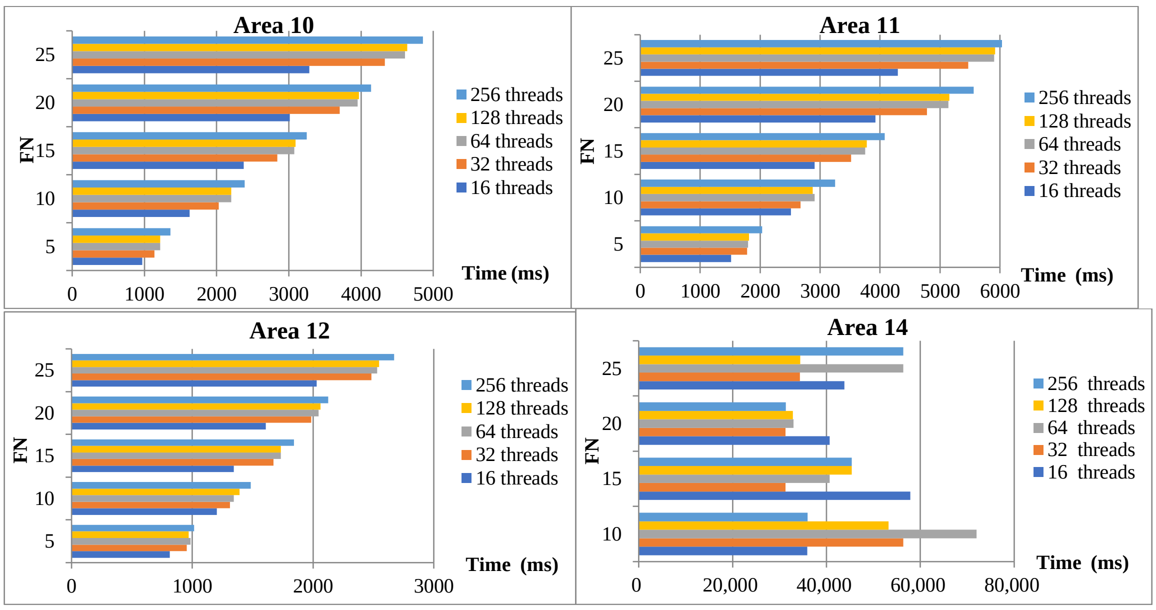


Figure 6. Time consumption of similarity computation in Area 10, Area 11, Area 12, and Area 14 under different combinations of *FN* and *TN*.

5.3. Algorithm Accuracy

Figure 7 delineates the matching accuracies of PSOM-GPU and the benchmark PRM in different road networks. Figure 8 shows the accuracy growth rate ($AGR_{(PSOM-GPU,PRM)}$) of PSOM-GPU compared with PRM.

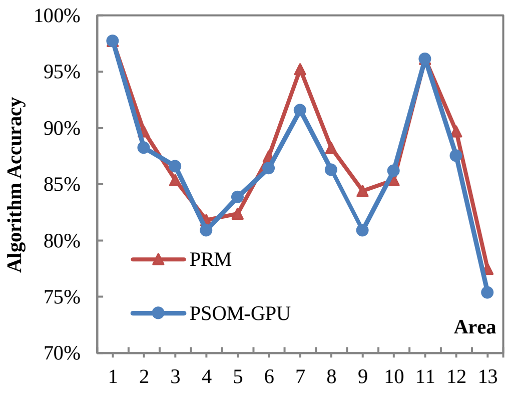


Figure 7. Algorithm accuracy *P* of PRM and PSOM-GPU in thirteen areas (Areas 1–13).

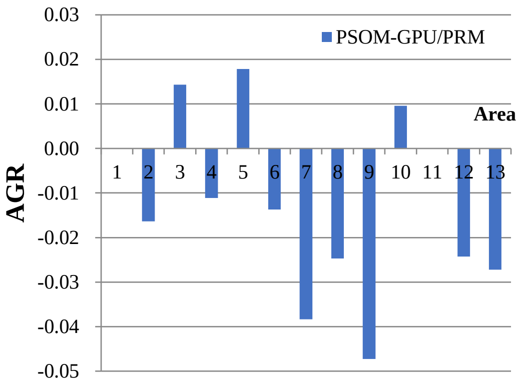
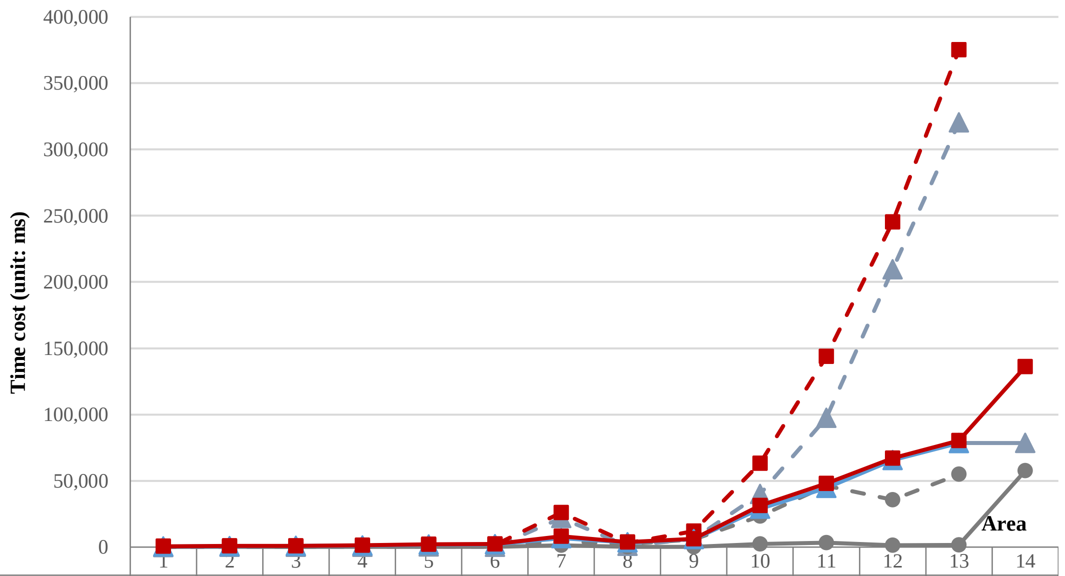


Figure 8. Accuracy growth rate (AGR) of PSOM-GPU and PRM.

The accuracies of PSOM-GPU are all greater than 75%, and more than half of the accuracies are greater than 85%, peaking at 97.73% in Area 1. The accuracies of PSOM-GPU do not change linearly when the number of nodes increases, but vary from region to region, following the same trend as the benchmark. As Figure 8 shows, the accuracy differences between PSOM-GPU and the benchmark are very small with an average $\mathrm{AGR}_{(\mathrm{PSOM\text{-}GPU},\mathrm{PRM})}$ of −0.012. For the areas with negative AGR, the absolute values are less than 0.05, which is in an acceptable accuracy error range. PSOM-GPU can achieve the same matching accuracy as the benchmark PRM, and correctly identify most matching-relationship (84.44% on average). In addition, since memory space required by PRM in Area 14 exceeds the tolerance value of the hardware, the accuracies of PRM are not calculated in this area. However, PSOM-GPU can successfully perform the matching process, and obtain an accuracy of 74.65%.

5.4. Algorithm Performance

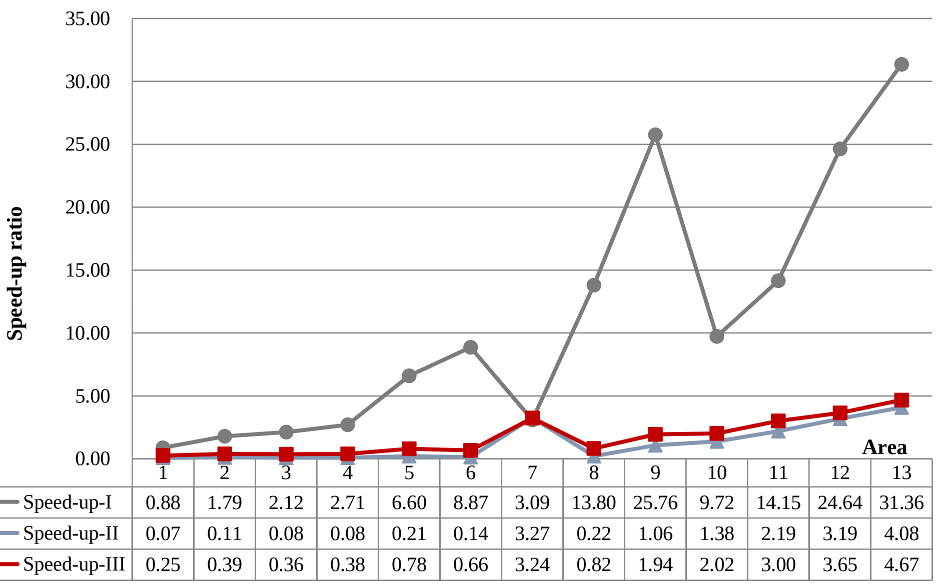
Figure 9 presents the specific time cost of the PSOM-GPU and the benchmark PRM. The time cost becomes larger with the increase in nodes. However, compared with the benchmark, the growth rate of time cost for PSOM-GPU is relatively flat. When the node number grows from hundreds to thousands, the time cost of the whole matching process based on PRM increases by 449 times from 141 ms in Area 1 to 63,312 ms in Area 10. The time cost for PSOM-GPU grows from 562 ms in Area 1 to only half of the benchmark with 31,407 ms in Area 10. The difference of time cost between the two methods is more obvious as the data scale increases. Besides, the PRM might not be able to get the road-network matching results with large amount of data (e.g., Area 14), while the PSOM-GPU performed successfully (e.g., spent 136,093 ms in Area 14). Overall, PSOM-GPU has a better performance than the benchmark in dealing with large-scale road-network matching.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14
PRM-I	110	250	265	422	1235	1250	4203	2360	5641	23,250	46,187	35,797	54,875	
PRM-II	31	78	63	94	391	297	21,828	797	6312	40,062	97,562	209,515	320,328	
PRM-III	141	328	328	516	1626	1547	26,031	3157	11,953	63,312	143,749	245,312	375,203	
PSOM-GPU-I	125	140	125	156	187	141	1360	171	219	2391	3265	1453	1750	57,593
PSOM-GPU-II	437	703	797	1219	1890	2187	6672	3672	5938	29,016	44,641	65,735	78,531	78,500
PSOM-GPU-III	562	843	922	1375	2077	2328	8032	3843	6157	31,407	47,906	67,188	80,281	136,093

Figure 9. The time cost of PRM and PSOM-GPU for road network matching. PRM-I, -II and -III denote the similarity computation, the matching-relationship identification and the global process, respectively, using PRM. PSOM-GPU-I, -II and -III denote the same three aspects, respectively, using PSOM-GPU.

To further evaluate the performance of PSOM-GPU, speed-up radio was calculated from local and global aspects (Figure 10). The local aspect focused on the performance improvement in the similarity computation stage and the matching-relationship identification stage. The global aspect embodied the performance of the whole process. Recorded execution time included the execution time on all stages of the algorithm, and the time spent on exchanging data. For the similarity computation stage, the time of reading data from disk was recorded, and for the matching-relationship identification stage, the time of writing data to disk was recorded. Besides, the recorded execution time for PSOM-GPU also contained the transfer time between the host memory and the GPU device memory. Thus, it is a fair and holistic comparison of the performance of different algorithms.



	1	2	3	4	5	6	7	8	9	10	11	12	13
● Speed-up-I	0.88	1.79	2.12	2.71	6.60	8.87	3.09	13.80	25.76	9.72	14.15	24.64	31.36
▲ Speed-up-II	0.07	0.11	0.08	0.08	0.21	0.14	3.27	0.22	1.06	1.38	2.19	3.19	4.08
■ Speed-up-III	0.25	0.39	0.36	0.38	0.78	0.66	3.24	0.82	1.94	2.02	3.00	3.65	4.67

Figure 10. The speed-up ratio of PSOM-GPU compared with PRM. I, II and III denote the three aspects of similarity computation, matching-relationship identification and global process, respectively.

From the local perspective, the speed-up ratios of both stages (the similarity computation and the matching-relationship identification) show an increasing trend as the number of road network nodes grows. The speed-up ratio of the similarity computation is increased from 0.88 in Area 1 to 31.36 in Area 13, and that of the matching-relationship identification grows from 0.07 to 4.08. The acceleration result of the algorithm is more obvious with increasing volumes of data. However, the speed-up performance of the two stages are not the same. Especially in Area 13, the speed-up ratio of similarity computation is 31.36, much higher than that of matching-relationship identification (4.08). The acceleration performance of similarity computation is better than the matching-relationship identification.

The roofline model [27] was adopted to analyze the theoretical performance of the two key processes (Figure 11). It is an intuitive visual performance model, providing the upper bound of performance and showing inherent hardware limitations. According to the specifications of NVIDIA GeForce GTX 960 GPU, the theoretical peak operational intensity of the GPU (hereafter, referred to as I_{max}) is 20.5 FLOP per Byte. The operational intensity of similarity computation is around I_{max}, reaching 22.46 FLOP per Byte in the average case, which means that the computing power of the GPU is fully utilized under similarity computation stage. The operational intensity of the matching-relationship identification stage is very low, only 2.5 FLOP per Byte. The memory bandwidth restricts the performance improvement of this stage. It indicates that the parallel strategy still has potential to be improved to increase the operational intensity of matching-relationship identification.

Note that there are some curve fluctuations in the stage of similarity computation in Figure 10. We can see the descending segments in the curve (speed-up-I) from Area 6 to Area 7 and from Area 9 to

Area 10, even though the data scale increases slightly. It is the reason that the time cost of the similarity function is highly dependent on the complexity of road and stroke. For instance, the average point number of a road segment is 3.6 in Area 6 while 7.8 in Area 7, and the average point number of strokes is 33.7 in Area 6 while 71.6 in Area 7. Thus, the road complexity of Area 7 is greater than that of Area 6, which leads to the lower speed-up in Area 7 than in Area 6. In addition, the speed-up ratio also differs greatly among the four different sub-functions in similarity-measure function. Thus, the speed-up of similarity computation not only depends on the data scale, but also relates with the complexity of the original road-network data and the corresponding function complexity.

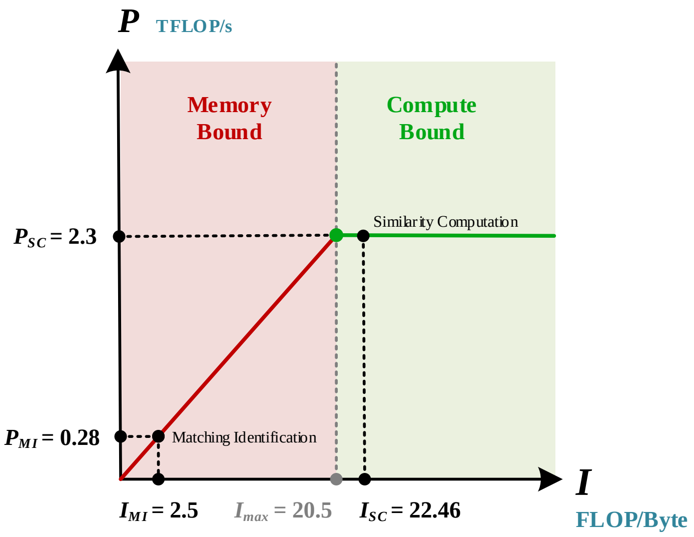


Figure 11. Roofline model for NVIDIA GeForce GTX 960 GPU with two stages of PSOM: similarity computation stage and the matching identification stage.

From the global perspective, the speed-up ratio is on the rise, increasing from 0.25 in Area 1 to 0.82 in Area 8, and then continues rising to more than 1, reaching 4.67 in Area 13. That is, when the data amount is small (e.g., 519 nodes in Area 8), less calculation is required, and the time is mainly spent on data transform. When facing with a large-size problem (e.g., 13,815 nodes in Area 13), the proposed parallelization strategy can significantly reduce the road-network matching time. Besides, the curve of the overall speed-up ratio is close to that of the matching-relationship identification process. This can be explained by the time-occupation ratio of the two key processes in PSOM (Figure 12). The time-consumption ratio of similarity computation is 0.15 on average, while it is 0.85 for matching-relationship identification. The matching-relationship identification process is the main consumption of the whole algorithm, which restricts the global performance improvement. More attention will be paid in parallel-matching identification stage in the future work.

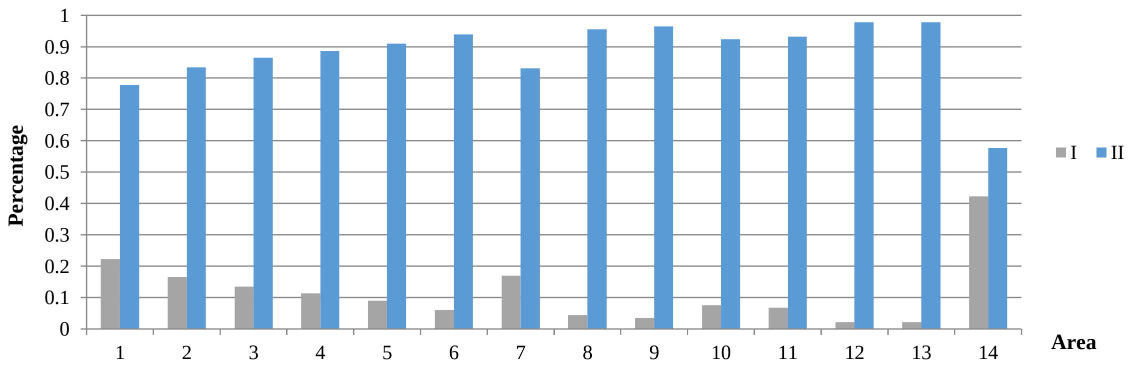


Figure 12. Time-occupation percentage of similarity computation (I) and matching-relationship identification (II).

In general, the proposed PSO-based parallel strategy of road network matching has a good performance when dealing with large amounts of data.

6. Conclusions

This study developed a parallel PSO-based road-network matching algorithm (PSOM) on graphics-processing units (GPU), which can be used to conquer the bottleneck problem of traditional road-network matching algorithms when facing big data. To fully use GPU threads, a data-partition and a task-partition strategy were proposed in similarity computation and matching-relationship identification stages, respectively. Fourteen pairs of road networks with different node scales were selected to verify the proposed PSOM method. Our results show that the road matching accuracy of the parallel GPU-accelerated PSOM approach achieved the same level as the benchmark PRM. When facing massive amounts of data, e.g. more than thousands of nodes, the proposed parallel PSOM method can significantly improve computational efficiency of the road-network matching process. Overall, the larger the data scale is, the more obvious the advantage of the PSOM-GPU approach in the speed-up of similarity computation and matching-relationship identification process is. According to these findings, the proposed parallel PSOM shows great potential for matching road-network in an effective and efficient manner.

This research also found that, although the parallel GPU-accelerated PSOM approach could promote the execution efficiency of both similarity computation and matching-relationship identification, the speed-up effects were different. It had better effect for the similarity computation stage. The memory bandwidth restricted the performance improvement of matching-relationship identification stage, which further affected the overall performance improvement. In the future, more work will be assigned for improving the speed-up performance of matching-relationship identification.

Author Contributions: Conceptualization, B.W. and S.Z.; Data curation, L.Y.; Formal analysis, B.W. and L.Y.; Funding acquisition, B.W.; Methodology, L.Y.; Project administration, B.W.; Resources, S.Z.; Software, L.Y.; Validation, W.Z. and D.W.; Visualization, D.W.; Writing—original draft, L.Y.; and Writing—review and editing, S.Z., R.W. and D.W.

Funding: This research was supported by the National Key Research & Development (R&D) Plan of China (No. 2017YFB050380504): and the State Key Laboratory of Geoinformation Engineering (grant number SKLGIE2017-M-4-1).

Acknowledgments: The authors would like to thank Qiao Xiong for his help in data processing during the experiment.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

The corresponding relationship between stroke structures is easier to determine than that of the node pairs. Thus, the common part of two matching stroke structure layers is extracted and employed as a reference. Each Pa_i may have more than one candidate nodes, thus similarities of one node Pa_i need multiple computations with the candidate node Pb_j in road network B. Two indicators were selected to measure initial probability: similarity of strokes that the node pairs individually belong to (denoted as Sim_{sk}) and similarity of reference distance evaluated by the distance between the node pairs and the starting point of stokes (denoted as Sim_{sd}). Assuming n is the stroke number the node Pa_i belongs to, the similarity between Pa_i and Pb_j is shown in Equation (A1).

$$Sim_{Pa_i,Pb_j} = \frac{\sum\limits_{k=1}^{n} \underset{s_{Pb_j}^m \in B_{cs}}{Max} \left[\varepsilon_1 (Sim_{sk}(s_{Pa_i}^k, s_{Pb_j}^m)) + \varepsilon_2 Sim_{sd}(Pa_i, Pb_j)\right]}{n} \tag{A1}$$

where $s_{Pa_i}^k$ represents the k_{th} stroke that Pa_i belongs to, and $s_{Pb_j}^m$ represents the m_{th} stroke that Pb_j belongs to. The number of strokes that one node belongs to may be more than one. For each node Pa_i, the similarity is evaluated by the average value of all strokes the node belongs to, and for each stroke $s^k_{Pa_i}$ of, the most similar stroke in the stroke set B_{CS} of Pb_j is selected. ε_1 is set as 0.6 and ε_2 is set as 0.4.

Sim_{sd} is the reference distance similarity, described in Equation (A2). ω_t is set as the reciprocal of error factor, which is represented as the average global distance between two maps, and d represents the reference distance evaluated by the distance between the node pairs Pa_i (or Pb_j) and the starting point of stokes $Pa_i^{\times}$ (or $Pb_j^{\times}$).

$$Sim_{sd} = \frac{1}{1 + \omega_t (d_{\{Pa_i, Pa_i^{\times}\}} - d_{\{Pb_j, Pb_j^{\times}\}})^2} \tag{A2}$$

Sim_{sk} is the stroke similarity, measured by geometrical similarity metrics. Here, *hausdorff* distance is chosen to measure geometrical similarity. Considering the stroke hierarchical relationship, if the current matching strokes are not in the initial set (i.e., $Layer \neq 1$), another two indices, the relative locational similarity of intersection Sim_{loc} and the relative direction similarity Sim_{dir}, are employed to measure the stroke similarity (Equation (A3)).

$$Sim_{sk} = \begin{cases} Sim_{sk} & (Layer = 1) \\ \varepsilon_1 Sim_{loc} + \varepsilon_2 Sim_{dir} + \varepsilon_3 Sim_{sk} & (Layer \neq 1) \end{cases} \tag{A3}$$

where ε_1 is set as 0.2, ε_2 is set as 0.4, and ε_3 is set as 0.4.

Sim_{loc} is measured by the relative position of the current stroke on the upper common stroke.

$$Sim_{loc} = \frac{1}{1 + \omega_t (loc_{s^k_{Pa_i}} - loc_{s^m_{Pb_j}})^2} \tag{A4}$$

Sim_{dir} is the relative direction similarity (Equation (A5)). The direction means the relative deviation angle between the upper common stroke and the current stroke. When the two strokes are intersected with multiple strokes in the upper layer, average direction similarity needs to be calculated.

$$Sim_{dir} = \frac{1}{1 + \omega_t (dir_{s^k_{Pa_i}} - dir_{s^m_{Pb_j}})^2} \tag{A5}$$

More details can be found in [13].

Appendix B

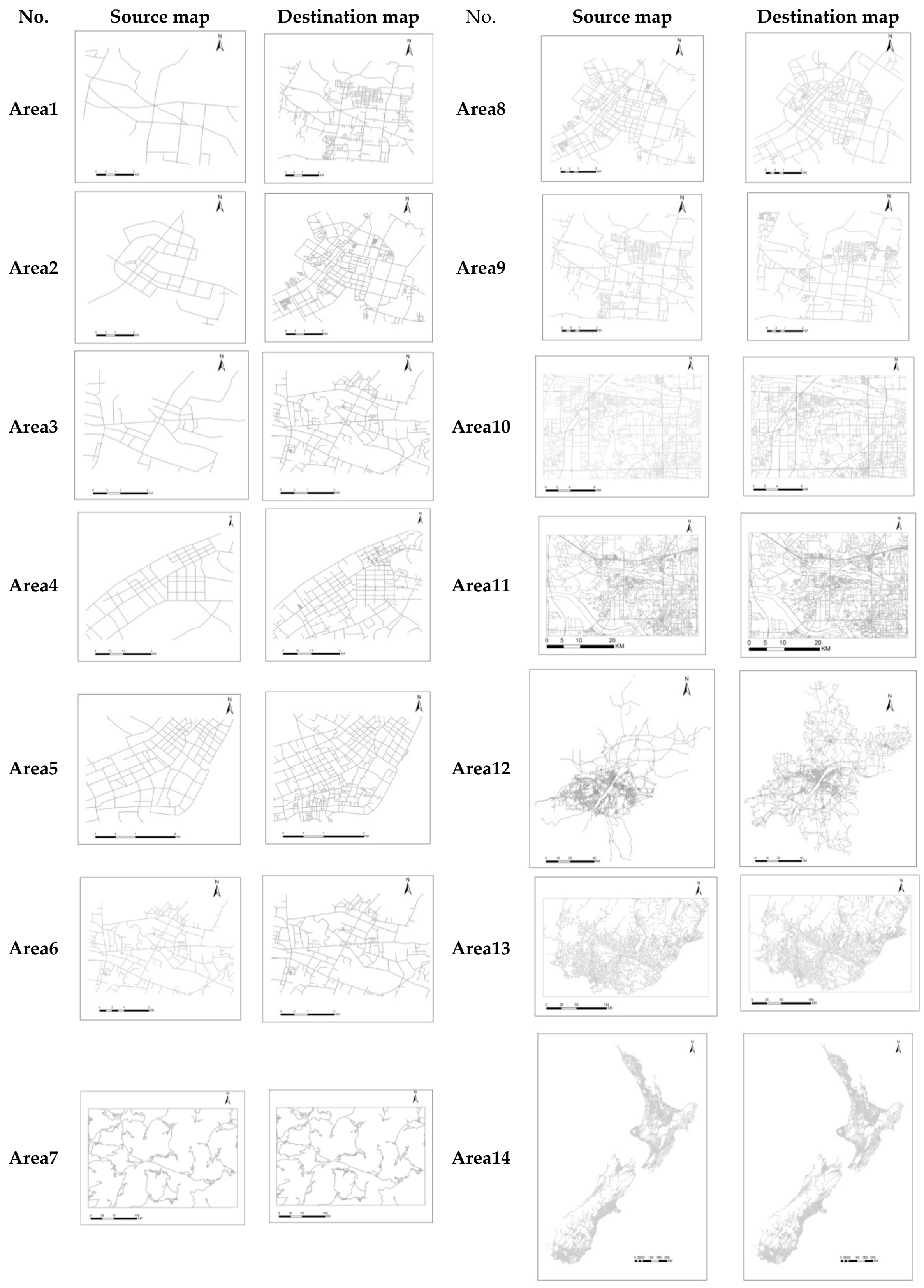


Figure A1. Road networks used for testing.

References

1. Li, L.; Goodchild, M.F. An optimisation model for linear feature matching in geographical data conflation. *Int. J. Image Data Fusion* **2011**, *2*, 309–328. [CrossRef]
2. Walter, V.; Fritsch, D. Matching spatial data sets: A statistical approach. *Int. J. Geogr. Inf. Sci.* **1999**, *13*, 445–473. [CrossRef]
3. Yang, B.; Luan, X.; Zhang, Y. A pattern—Based approach for matching nodes in heterogeneous urban road networks. *Trans. GIS* **2014**, *18*, 718–739. [CrossRef]
4. Beeri, C.; Kanza, Y.; Safra, E.; Sagiv, Y. Object fusion in geographic information systems. In Proceedings of the Thirtieth International Conference on Very Large Data Bases-Volume 30, Toronto, ON, Canada, 31 August–3 September 2004; Morgan Kaufmann: Toronto, ON, Canada, 2004; pp. 816–827.
5. Song, W.; Keller, J.M.; Haithcoat, T.L.; Davis, C.H. Relaxation—Based point feature matching for vector map conflation. *Trans. GIS* **2011**, *15*, 43–60. [CrossRef]
6. Safra, E.; Kanza, Y.; Sagiv, Y.; Doytsher, Y. Ad hoc matching of vectorial road networks. *Int. J. Geogr. Inf. Sci.* **2013**, *27*, 114–153. [CrossRef]
7. Min, D.; Zhilin, L.; Xiaoyong, C. Extended hausdorff distance for spatial objects in gis. *Int. J. Geogr. Inf. Sci.* **2007**, *21*, 459–475. [CrossRef]
8. Tong, X.; Liang, D.; Jin, Y. A linear road object matching method for conflation based on optimization and logistic regression. *Int. J. Geogr. Inf. Sci.* **2014**, *28*, 824–846. [CrossRef]
9. Zhang, M.; Meng, L. Delimited stroke oriented algorithm-working principle and implementation for the matching of road networks. *Geogr. Inf. Sci.* **2008**, *14*, 44–53. [CrossRef]
10. Xiong, D.; Sperling, J. Semiautomated matching for network database integration. *ISPRS J. Photogramm. Remote Sens.* **2004**, *59*, 35–46. [CrossRef]
11. Gabay, Y.; Doytsher, Y. Automatic adjustment of line maps. In Proceedings of the GIS/LIS '94, Phoenix, AZ, USA, 25–27 October 1994; pp. 191–199.
12. Cueto, K. A feature-based approach to conflation of geospatial sources. *Int. J. Geogr. Inf. Sci.* **2004**, *18*, 459–489.
13. Yang, L.; Wan, B.; Wang, R.; Zuo, Z.; An, X. Matching road network based on the structural relationship constraint of hierarchical strokes. *Geomat. Inf. Sci. Wuhan Univ.* **2015**, *40*, 1661–1668.
14. Zhao, D.; Sheng, Y. Research on automatic matching of vector road networks based on global optimization. *Acta Geod. Cartogr. Sin.* **2010**, *39*, 416–421.
15. Zhang, Y.; Yang, B.; Luan, X. Automated matching crowdsourcing road networks using probabilistic relaxation. *Acta Geod. Cartogr. Sin.* **2012**, *41*, 933–939. [CrossRef]
16. Luan, X. A structure-based approach for matching road junctions with different coordinate systems. In Proceedings of the Twenty-Second ISPRS Congress, Melbourne, Australia, 25 August–1 September 2012; pp. 41–46.
17. Liu, C.; Qian, H.; Xiao, W.; Haiwei, H.E.; Chen, J. A linkage matching method for road networks considering the similarity of upper and lower spatial relation. *Acta Geod. Cartogr. Sin.* **2016**, *45*, 281–286.
18. Fan, H.; Yang, B.; Zipf, A.; Rousell, A. A polygon-based approach for matching openstreetmap road networks with regional transit authority data. *Int. J. Geogr. Inf. Sci.* **2016**, *30*, 748–764. [CrossRef]
19. Zhang, J.; Wang, Y.; Zhao, W. An improved probabilistic relaxation method for matching multi-scale road networks. *Int. J. Dig. Earth* **2018**, *11*, 1–21. [CrossRef]
20. Luu, K.; Noble, M.; Gesret, A.; Belayouni, N.; Roux, P.F. A parallel competitive particle swarm optimization for non-linear first arrival traveltime tomography and uncertainty quantification. *Comput. Geosci.* **2018**, *113*, 81–93. [CrossRef]
21. Yang, B.; Zhang, Y.; Luan, X. A probabilistic relaxation approach for matching road networks. *Int. J. Geogr. Inf. Sci.* **2013**, *27*, 319–338. [CrossRef]
22. Tong, X.; Shi, W.; Deng, S. A probability-based multi-measure feature matching method in map conflation. *Int. J. Remote Sens.* **2009**, *30*, 5453–5472. [CrossRef]
23. Finch, A.M.; Wilson, R.C.; Hancock, E.R. *Matching Delaunay Triangulations by Probabilistic Relaxation*; Springer: Berlin/Heidelberg, Germany, 1995; pp. 350–358.

24. Hasan, S.; Bilash, A.; Shamsuddin, S.M.; Hassanien, A.E. Gpu-based capso with n-dimension particles. In Proceedings of the International Conference on Advanced Machine Learning Technologies and Applications, Cairo, Egypt, 22–24 February 2018; pp. 459–467.
25. Dali, N.; Bouamama, S. Gpu-pso: Parallel particle swarm optimization approaches on graphical processing unit for constraint reasoning: Case of max-csps. *Procedia Comput. Sci.* **2015**, *60*, 1070–1080. [CrossRef]
26. Dali, N.; Bouamama, S. Different parallelism levels using gpu for solving max-csps with pso. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation, San Sebastian, Spain, 5–8 June 2017; pp. 2070–2077.
27. Williams, S.; Waterman, A.; Patterson, D. Roofline: An insightful visual performance model for multicore architectures. *Commun. ACM* **2009**, *52*, 65–76. [CrossRef]