International Journal of Geo-Information

MDPI

Article

A Novel *k*-Means Clustering Based Task Decomposition Method for Distributed Vector-Based CA Models

Zhenqiang Li [1], Xuefeng Guan [1,2,*], Huayi Wu [1,2,] and Jianya Gong [1]

[1] The State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, 129 Luoyu Road, Wuhan 430079, China; lizq@whu.edu.cn (Z.L.); wuhuayi@whu.edu.cn (H.W.); gongjy@whu.edu.cn (J.G.)

[2] Collaborative Innovation Center of Geospatial Technology, 129 Luoyu Road, Wuhan 430079, China

* Correspondence: guanxufeng@whu.edu.cn; Tel.: +86-27-6877-8311

Academic Editor: Wolfgang Kainz

Received: 5 February 2017; Accepted: 22 March 2017; Published: 23 March 2017

Abstract: More and more vector-based cellular automata (VCA) models have been built to leverage parallel computing to model rapidly changing cities and urban regions. During parallel simulation, common task decomposition methods based on space partitioning, e.g. grid partitioning (GRID) and recursive binary space partitioning (BSP), do not work well given the heterogeneity of VCA parcel tasks. In this paper, to solve this problem, we propose a novel task decomposition method for distributed VCA models based on *k*-means clustering, named KCP. Firstly, the polygon dataset is converted into points based on centroids, which combines the size of two parcels and the outer distance. A low-cost recursive quad-partition is then applied to decide the initial cluster centers based on parcel density. Finally, neighbor parcels can be allocated into the same subdivision through *k*-means clustering. As a result, the proposed KCP method takes both the number of tasks and computing complexity into consideration to achieve a well-balanced local workload. A typical urban VCA growth model was designed to evaluate the proposed KCP method with traditional spatial partitioning methods, i.e. GRID and BSP. KCP had the shortest total simulation time when compared with GRID and BSP. During experimental urban growth simulations, the time spent on a single iteration was reduced by 15% with the BSP and by 25% with the GRID method. The total simulation time with a 120 m neighborhood buffer size was reduced by more than one hour to around three minutes with 32 cores.

Keywords: vector-based CA; distributed computing; *k*-means clustering; task decomposition

1. Introduction

Vector-based cellular automata (VCA) models extend traditional raster-based cellular automata (raster-based CA) models by using irregular vector polygons to represent actual geographic features, e.g. parcels and blocks [1]. This extension relieves the sensitivity of a CA model to spatial resolution and breaks the limits on uniform neighborhood definition [2,3]. VCA models have been widely used to simulate urban dynamics, e.g. land-use and land-cover changes, urban growth [4–6], urban planning [7], etc. However, large-scale and highly detailed VCA/CA models usually require massive computing resources to obtain timely simulation results. Thus, researchers are turning to parallel computing to accelerate these time-consuming model simulations, e.g. programs like pRPL [8,9], pSLEUTH [10], CAMEL [11].

In a given VCA model, irregular parcel polygons evolve through a number of discrete time steps following a set of transition rules based on the states of neighboring parcels. Thus, a basic parallel VCA (pVCA) task is equivalent to the corresponding polygon parcel. The computation of

one pVCA task includes neighbor search and parcel status transition, and the amount is generally proportional to the area of a parcel.

The task decomposition in pVCA models can be formulated as typical space partitioning problems. There is a growing collection of space partitioning algorithms, which can be grouped according to how partitioning is conducted and classified into two categories; flat grid partitioning and hierarchical tree partitioning. Flat grid-based partitioning algorithms use a rectangular grid to partition the target space, thus directly obtaining discrete regular subdivisions. Hierarchical tree-based algorithms on the other hand, divide the target space into two or more disjoint subsets recursively, eventually producing a binary space partitioning (BSP) tree, e.g., KD-tree.

However, both flat grid-based and hierarchical tree-based algorithms are not easily applicable to pVCA task decomposition because neither the equal subdivision area nor equal task number can guarantee an equal workload. Flat grid partitioning can produce equal-area subdivisions, but the irregular shape of pVCA parcels leads to different task numbers for each subdivision. Conversely, hierarchical tree partitioning can easily produce equal task numbers by recursive division, but the workload of each subdivision is still unbalanced because of the varying computing complexity of pVCA tasks.

In this paper, a novel pVCA task decomposition algorithm based on general *k*-means clustering named KCP is proposed to overcome these drawbacks. The KCP method formulates a customized *k*-means clustering to cluster tasks with higher geographical proximity. This method uses parcel centroid to represent parcel polygons and defines a proximity distance combining parcel size and outer distance. Through an iterative process, the generated subdivisions will contain neighboring parcels, while, at the same time, the technique will separate large-sized parcels using the centroid distance. In this way, the KCP decomposition can take both task numbers and their computation complexity into consideration to obtain better workload balance. Since the workload of each subdivision depends on the number of allocated tasks, the computing complexity and communication overhead depend on the amount of ghost parcels, the NSD-PA (normalized standard deviation of the area of parcels), NSD-PN (normalized standard deviation of the number of parcels), and the number of total ghost agents that are applied to indicate local workload balance and the communication overhead.

An additional VCA-based urban growth model was developed to evaluate the efficiency of the proposed KCP algorithm. Further, we parallelized the model based on a bulk synchronous parallel model and adopted the ghost agent strategy to reduce information exchange frequency. Two groups of experiments were designed, with four subdivision sizes, 4, 8,16, and 32, to test their scalability and three buffer sizes, 120 m, 240 m, and 360 m, to test their effects on the communication overhead. In addition, we made a detailed study on the local workload and communication overhead separately using the NSD-PN, NSD-PA, and the number of total ghost agents. The experimental results show that KCP employs the least local computing time with acceptable communication overhead and achieves the best parallel performance, compared with two common decomposition methods, GRID and BSP. The proposed KCP method can be used to partition spatial tasks of large-scale detailed VCA models effectively, which can shorten the computing time and improve the efficiency of adopting a VCA model.

The rest of the paper is outlined as follows. Section 2 provides a detailed introduction to VCA models and an overview of existing task decomposition methods with spatial-partitioning. Section 3 explains the proposed KCP task decomposition method. In Section 4, a parallel VCA urban growth model is presented. In Section 5, we evaluate and compare the KCP performance to the GRID and BSP performance. In Section 6, we present a discussion and draw some conclusions.

2. Background and Related Work

2.1. VCA Models

VCA originates from the idea of using irregular polygons instead of regular cells to represent real features in simulation models [1,2]. A number of researchers have implemented this idea with

various spatial segregation strategies, e.g. resolution elements [3], Voronoi polygons [4], Delaunay triangles, and cadastral parcel polygons [5–8]. Among these approaches, parcel polygons, rather than a mixture of regular cells, most closely correspond to features in the real world. Thus, irregular representation of VCA models provides the capability to model more detailed and complex behaviors and interaction between features.

In contrast to the Moore or Von Neumann neighborhood definitions of raster-based CA models, a VCA neighborhood can be defined by a spatial relationship; adjacency or buffer. In a neighborhood, two parcel polygons sharing points or edges are considered adjacent neighbors. A buffer neighborhood is defined by the distance between parcel polygons. All the polygons within a user-defined buffer distance are considered neighbors. Dahal et al. [9] made a detailed study of the neighborhood sensitivity in irregular CA models using nine types of neighborhoods and buffer distances.

The transition functions in VCA models are usually derived from GIS suitability analysis, e.g. Neighborhood, Accessibility, Suitability, and Zone status (NASZ) [10]. However, each parcel polygon contains a different number of neighbors, and they are also different in size and shape. These heterogeneities make the computing cost of each VCA task fluctuate very sharply.

Compared to raster CA, VCA model simulations are, however, computer-intensive because of irregular spatial representation, complex transition rules, and massive input polygons [8,11]. Following parallel CA models, parallel computing technology is required to support large scale detailed VCA model simulation. During parallelization, task decomposition is critical to fully utilize computing and storage resources.

2.2. Task Decomposition Methods in Parallel VCA Models

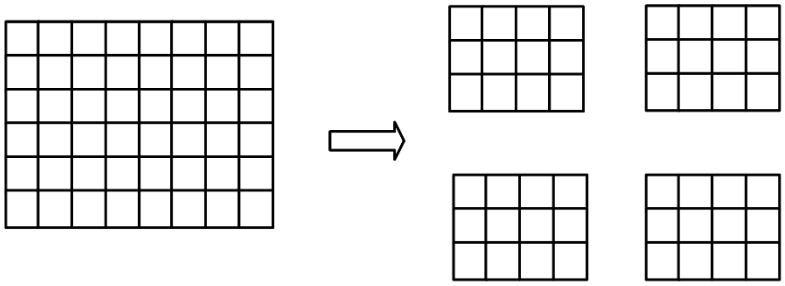
The spatial task decomposition method determines the workload balance and communication overhead influencing the model parallel computing efficiency. The spatial tasks in pVCA models are mutually dependent as the status of neighbors is involved in each polygon transition. Moreover, the shape and size of corresponding polygons in a pVCA task are very different, and this cannot guarantee that equal task numbers or equal subdivision areas produce equal local workloads. Thus, pVCA task decomposition should firstly address task dependence to minimize the communication overhead and then take both task computing complexity and task number into consideration to obtain a balanced workload.

In practice, space partitioning strategies are usually used to conduct a near optimal pVCA task decomposition. Space partitioning divides the target space into a set of non-overlapping subdivisions, and then VCA polygons can be allocated to different subdivisions by their spatial relationships. Space partitioning algorithms can be categorized into flat grid-based methods and hierarchical tree-based methods according to the data structure applied in the partitioning process.

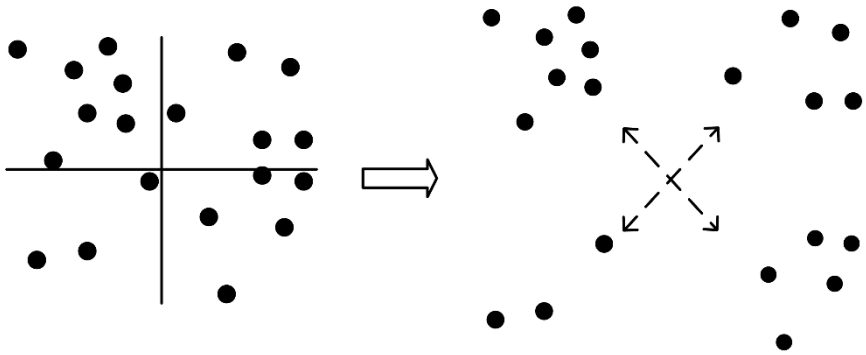
2.2.1. Flat grid-Based Methods

Flat grid-based methods are widely adopted in raster-based CA models [12–15]. These methods usually exhibit three forms; column-wise, row-wise, and grid-wise. The established grid firstly partitions the target space into flat regular cells and directly allocates simulation tasks to the corresponding cells. Several parallel CA libraries support these partition methods, e.g. pRPL, pRPL2, and CAMEL.

Figure 1 shows an example of a grid-wise partition applied in a raster-based CA(1a) and in a VCA model(1b). Since raster-based CA models contain equal-sized uniform cells, task decomposition with this type of space partitioning can easily achieve both a balanced workload and a minimum communication overhead. However, when it is applied to VCA models, these methods will lead to an ill-balanced workload because of irregular polygons and heterogeneous distribution; hierarchical tree-based methods such as BSP overcome this limitation by recursively bisecting tasks.



(a) 2x2 grid-wise partitioning of raster-based CA

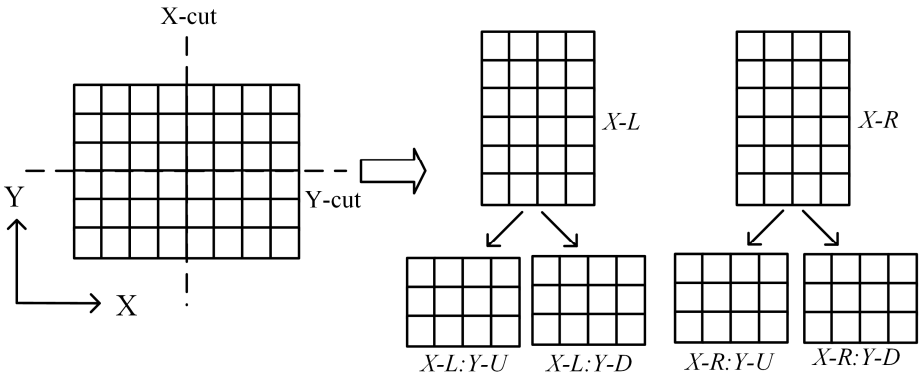


(b) 2x2 grid-wise partitioning of VCA

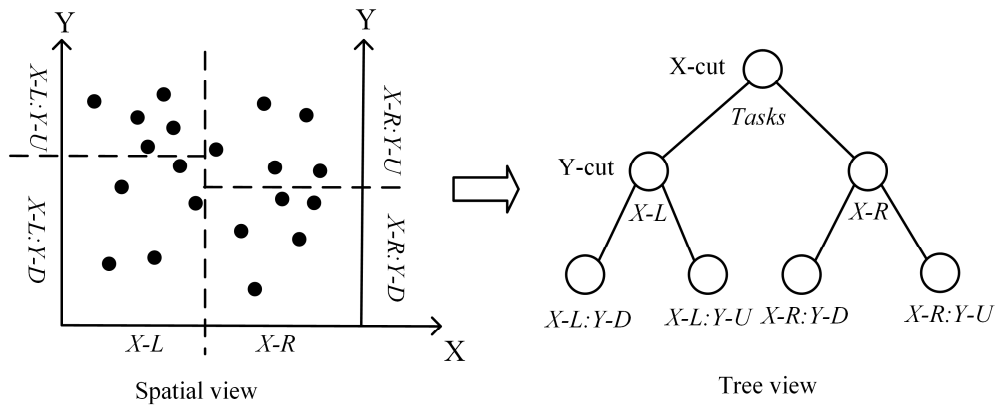
Figure 1. The grid-wise partitioning method of (**a**) raster-based cellular automata (CA); and (**b**) vector-based cellular automata (VCA).

2.2.2. Hierarchical Tree-Based Methods

Figure 2 illustrates the hierarchical space partitioning process by a binary space partitioning tree. When applied in pVCA task decomposition, the BSP tree can produce subdivisions with an equal number of tasks. However, the BSP tree method does not consider the task computing complexity and still cannot guarantee workload balance. In addition, BSP lacks a strong guarantee on the communication overhead, which makes it degenerate when exploring neighborhood configurations.



(a) spatial tasks partitioning with a BSP tree in raster-based CA



(b) spatial tasks partitioning with a BSP tree in VCA

Figure 2. Spatial tasks partitioning with a binary space partitioning (BSP) tree of (**a**) raster-based CA and (**b**) VCA.

3. The *k*-Means Based Partitioning Method

3.1. k-Means Clustering Method

The *k*-means clustering method is one of the most widely used clustering algorithms because of its simplicity, efficiency, and empirical success [16]. Given a set of samples (x_1, x_2, …, x_n) in which each sample is a R^d real vector, *k*-means clustering partitions *n* samples into *k* sets $S = \{S_1, S_2, \ldots, S_k\}$, where ($k \le n$) and minimizes the within-cluster sum of squares (WCSS), i.e. the sum of the distance functions of each point within one cluster of the *k* center, as in Equation (1),

$$\arg\min_{s} \sum_{i=1}^{k} \sum_{X \in S_i} \| X - \mu_i \|^2 \quad (1)$$

where μ_i is the cluster center in S_i. The kernel of *k*-means clustering is the user-defined distance function, which provides a quantitative measurement of proximity between features. The basic *k*-means clustering algorithm has been extended in many different ways, including soft membership in Fuzzy *c*-means [17,18], recursively hierarchical divisive bisecting of the *k*-means [19], automatically critical based *k* finding in X-means [20], etc. Readers can refer to [16] for a more detailed review.

3.2. The KCP Task Decomposition Method

In VCA models, both adjacent and buffer neighborhood definitions follow a common idea that closer features have a more important effect on the center feature. Inspired by this approach, the proposed KCP method formulates a new *k*-means clustering model to decompose VCA tasks. The KCP method consists of three phases; mapping, initial centers choosing, and clustering. The mapping phase converts polygon-based tasks into point types. The second phase chooses initial centers for each future cluster. The clustering phase carries out iterative clustering computation.

(1) Mapping phase

As the *k*-means clustering algorithm requires point input, a mapping is firstly conducted to convert VCA polygons into points representing the centroid of its minimum bounding box. Shown in Figure 3, the distance *d* between polygons consists of three parts: d_1 and d_3 represent the sizes of two polygons, and d_2 is the actual outer spatial distance between parcels. By this distance definition, two bigger parcels are more likely to be separated. For example, parcel *A* and *B* are big parcels, and parcel *C* is small. Although *A* and *B* have shorter outer distances than *A* and *C*, nevertheless, *A* and *C* are more likely to be allocated into the same subdivision. Furthermore, *A* and *B* are more likely to

be allocated into different subdivisions because *A* and *C* have a smaller combination distance than *A* and *B* under this mapping strategy.

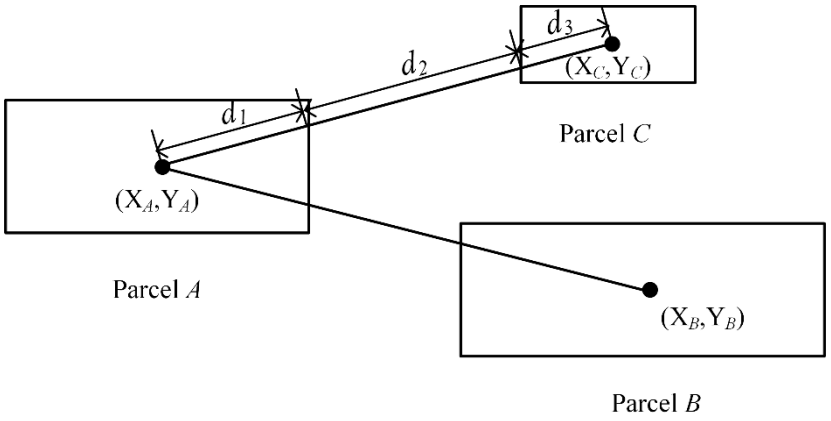


Figure 3. An illustration of distance definition between parcels.

Thus, this mapping strategy exhibits two advantages: (1) two close big polygons may end up with a large distance, and thus these two polygons are likely to be allocated into two different subdivisions and (2) though two dispersed small polygons have share a smaller distance, they are likely to be allocated into the same subdivisions. Since bigger polygons usually create more computation than smaller ones when geometry computation involved, these two traits can ensure a more balanced local workload among subdivisions.

(2) Choosing the k initial cluster centers

The total number of clusters, *k*, and initial centers are important for convergence speed and final clustering results. There are many customizations designed to find a cluster number or initial centers, such as canopy clustering and hierarchical clustering. As for task decomposition, *k* can be trivially defined as the number of processors. Usually initial cluster centers should be picked from the high-density areas of input points. In this paper, a simple recursive 4-way method was designed to determine the initial cluster centers. Shown in Figure 4, it partitions the whole space into 4^{level} sub grids and records the point number in each sub grid. For a given cluster number *k*, the geometric centers with the *k* highest values are chosen as the initial cluster centers.

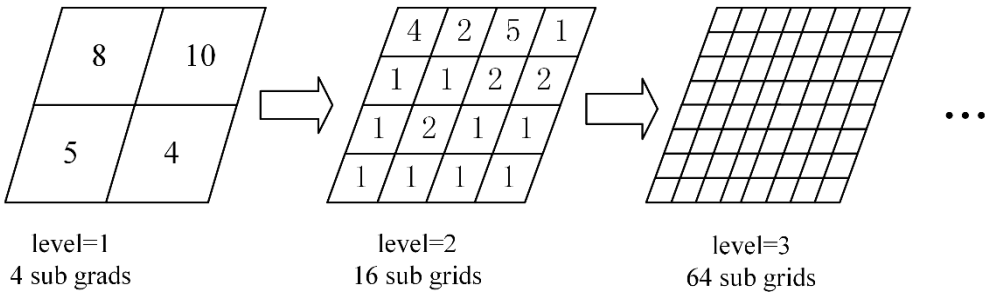


Figure 4. An illustration of recursive 4-way partitioning.

In order to determine the partitioning level, let $\partial \geq 1$ indicate the selection freedom factor (SFF); then the least total sub grid numbers is $n = \partial k$ and the least partition level is $level = \lceil \log_4^n \rceil$. For example, *k* is 4, ∂ is 4, and the minimum number of sub grids is 16, which means the minimum partition level is 2.

(3) Clustering phase

Each subdivision has a center at C_j, which represents the geometric center of all the spatial tasks belonging to this subdivision. The goal of *k*-means clustering is to minimize the value J in Equation (2).

$$J = \sum_{i=1}^{n} \sum_{j=1}^{k} pred_{i \to j} \left\| X_i - C_j \right\| \tag{2}$$

where $pred_{i->j}$ is a criterion function. If the feature X_i belongs to the subdivision C_j, the $pred_{i->j}$ is 1; otherwise it is 0. Repeat the following three steps until the cluster members stabilize.

(a) Generate a new clustering by assigning each spatial task to its closest cluster center according to Equation (3).

$$c_i^t := \arg\min \left\| X_i - C_j \right\| \tag{3}$$

(b) Compute new cluster centers following Equation (4)

$$C_j^{t+1} = \frac{\sum pred_{i->j} X_i}{\sum pred_{i->j}} \tag{4}$$

(c) Check if the current clustering results are stable according to the criteria in Equation (5). If true, go to step a) and repeat the assignment operation; otherwise the clustering is finished.

$$\forall_j \left\| C_j^{t+1} - C_j^t \right\| > \theta \tag{5}$$

4. A Parallelized Urban Growth Model Based on VCA

In order to evaluate the KCP method, a parallel urban growth model was designed to study the performance of KCP on workload-balance and communication overhead.

4.1. The VCA Based Urban Growth Model

The design of the VCA model is illustrated in Figure 5. The model input contains a collection of data layers related to urban growth, including transportation, terrain, and parcels. The model output represents a future urban growth area. The whole urban growth simulation can be viewed as an iteration process; each iteration spans a period of time, e.g. a month, a quarter, or a year. During iteration, the new status of each polygon is derived according to the status and its neighbor information.

In this model, each parcel polygon has two statuses: undeveloped and developed. The initial parcels status of each polygon was extracted from the initial urban map by overlap analysis. The polygons covered by the initial urban area were assigned to developed parcels; otherwise they were set as undeveloped. A buffer neighborhood was included in the neighbor definition, and three buffer distances (120 m, 240 m, and 360 m) were applied to evaluate how the neighborhood size affected the communication overhead.

As shown in Figure 6, the transition status of a parcel contains two phases; calculating the transition probability to developed status and conducting a transition rule test. The NASZ scheme was used to calculate the transition probability of the parcels. This scheme contains three steps; accumulate the effect of neighbors, evaluate the suitability of a parcel on its own, and synthesize these two results.

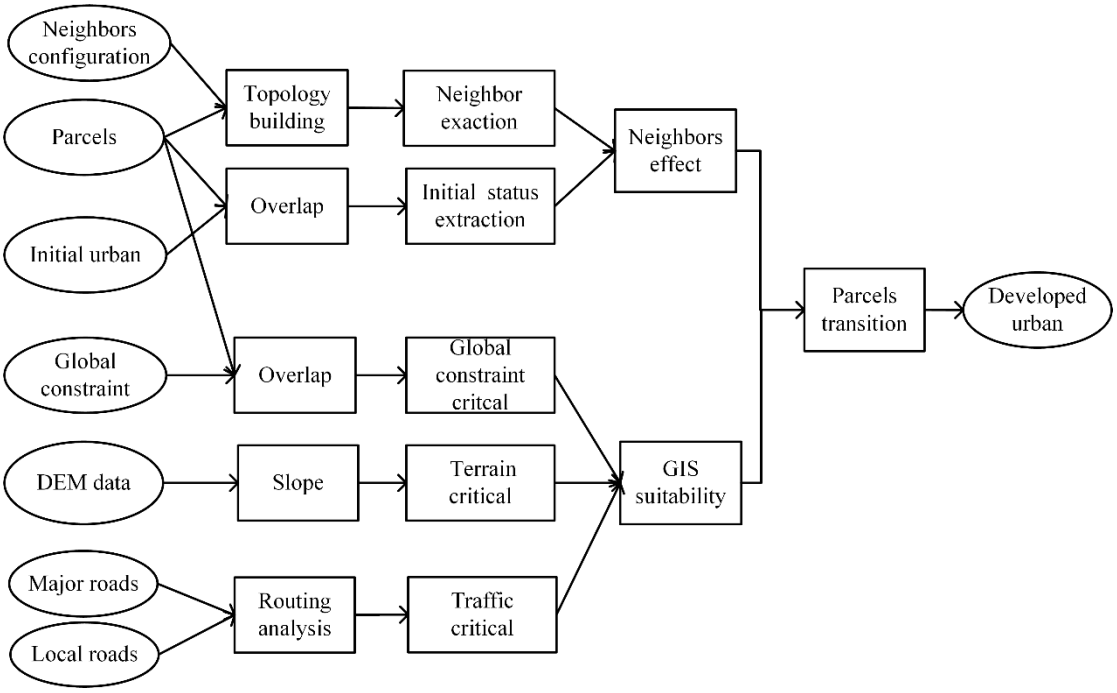


Figure 5. A typical urban growth model based on GIS suitability analysis.

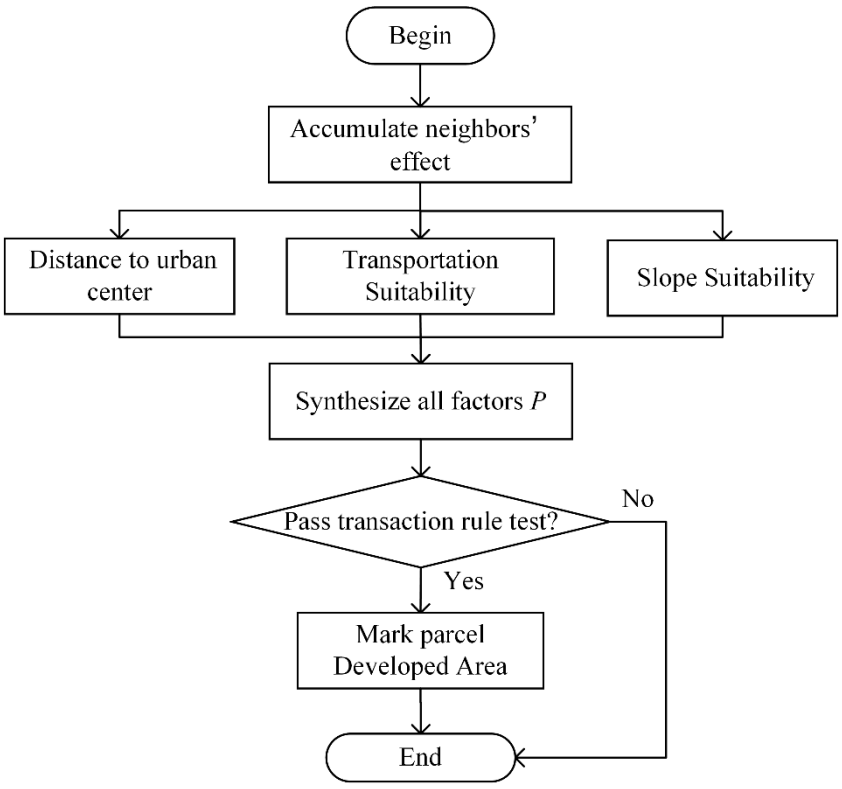


Figure 6. The evolution flowchart of each parcel polygon.

The derived transition probability combines its current conditions, the status of its neighbors, and a set of suitability criteria in Equation (6),

$$p_i^t = F_g cond(c_i^{t-1}) \sum_j effect(c_j^{t-1}) \quad (6)$$

where, C_i^t is the current condition of parcel i at time $t-1$ and F_g is the global factor indicating the external driving force, such as positive policy, the vitality of the city, etc. The second term $cond(C_i)$ is

a function that calculates the suitability of a parcel for development, based on a set of suitability criteria, including the distance to major roads, the slope, and area. The last accumulated term represents the accumulated effect of neighboring parcels on the center parcel, e.g. if there are more developed cells around it, it is more likely to be developed.

After computing transition probability, a transition rule test is applied to all parcels. Generally, there are two kinds of transition rule tests, i.e. threshold and stochastic tests. The threshold type uses a user-specified threshold value to determine if the transition probability of a parcel can pass the test. In contrast, the stochastic type of rule test applies a random number generator to determine if a parcel can make a transition. In this way, the stochastic type of test can introduce random factors to an urbanization process, which can simulate uncertainties such as policy planning, environment change, etc. A threshold transition rule was applied in our urban growth model parallelization testing design.

4.2. Research Area and Input Datasets

The research area covers a small city near San Bernardino, California, with area of around 207.823 km^2. The major parcel dataset used in this case study was downloaded from the Internet [21] and contains 38820 parcels, as shown in Figure 7.

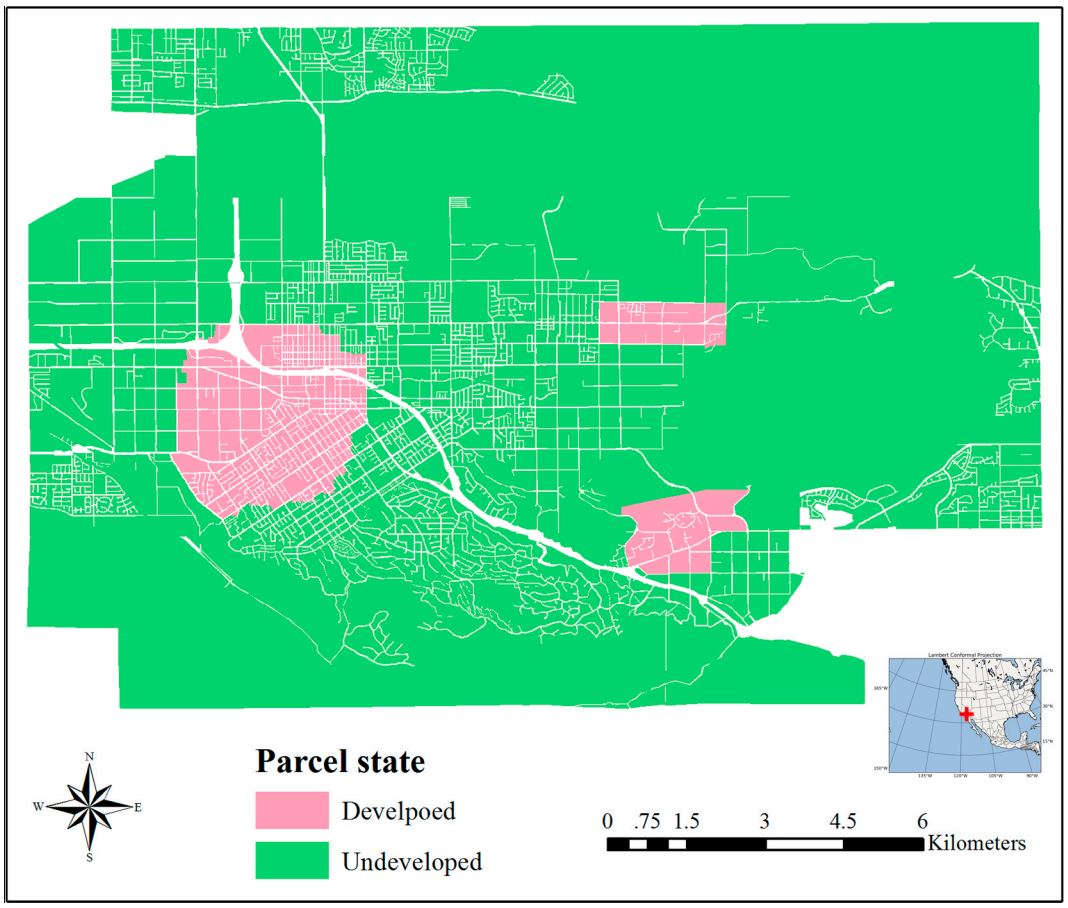


Figure 7. The reasearch area and experimental parcel dataset.

The detailed attributes of other relevant datasets are also listed in Table 1.

Table 1. Basic information of the dataset used in the VCA-based urban growth model.

Dataset	Format	Scale	Description
Parcels	Vector	1:2500	Parcels represented as polygons, including their areas, owner types
Initial urban Area	Vector	1:2500	Initial urban areas, the closer the undeveloped parcels are to these areas, the larger probability to be developed
Roads	Vector	1:2500	Roads including local roads and major roads. The parcel accessibility factor was derived from the roads layer
DEM	Raster	1:5000	The DEM of the research area where slope image can be extracted

4.3. Parallelization of the VCA-Based Urban Growth Test Model

As described in Section 4.1, the kernel of an urban growth model is to evaluate the new status of each parcel iteratively, and each parcel is equal to an independent task unit in the whole simulation process. During parallelization, the task decomposition method, e.g. KCP, GRID, or BSP, divides all VCA tasks into discrete groups, and each processing element is in charge of one group simulation.

A bulk synchronous parallel strategy was used to coordinate the computation and communication during parallel simulation [22], consisting of local computing and synchronization in each iteration. In the local computing phase, the local status of each parcel was computed according to the transition function. The synchronization phase conducts neighboring information exchange between processors. The parallel VCA urban growth model is illustrated in Figure 8.

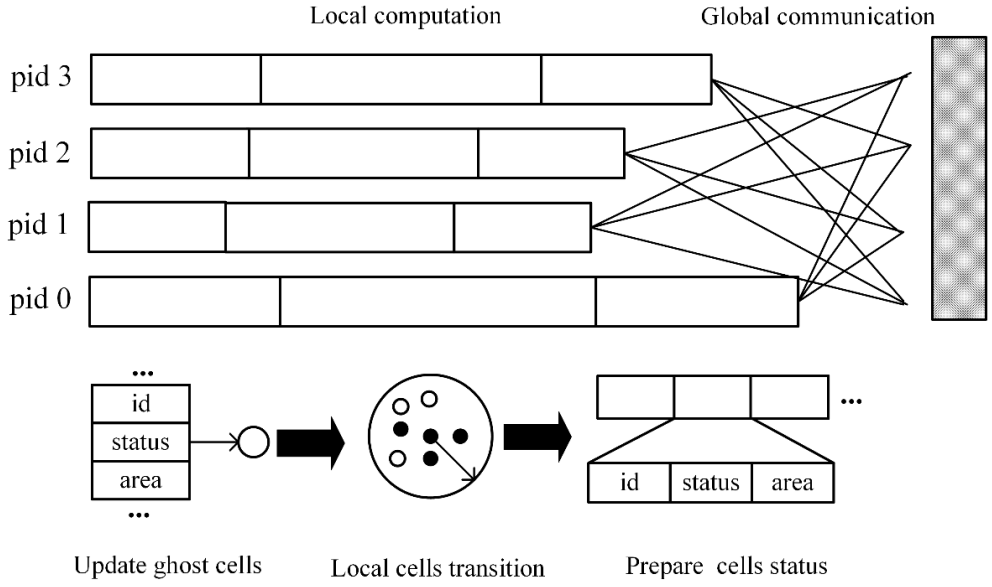


Figure 8. Parallelization of the urban growth model based on a bulk synchronous parallel strategy.

Neighboring information exchanges result in additional communication overhead. In order to reduce the communication overhead, ghost parcels [23,24] were created to lower the status exchange frequency between cells. Figure 9 illustrates how ghost parcels reduce the communication overhead [14]. To avoid requesting the status of remote parcels during each iteration, current processing nodes will cache selected parcels on the local process; these are called 'ghost parcels'. Each ghost parcel acts as a copy of an original remote parcel. Local process can query the status of a ghost parcel but does not have permission to update their status. Their status can be only updated during the synchronization phase when the original parcels are changed.

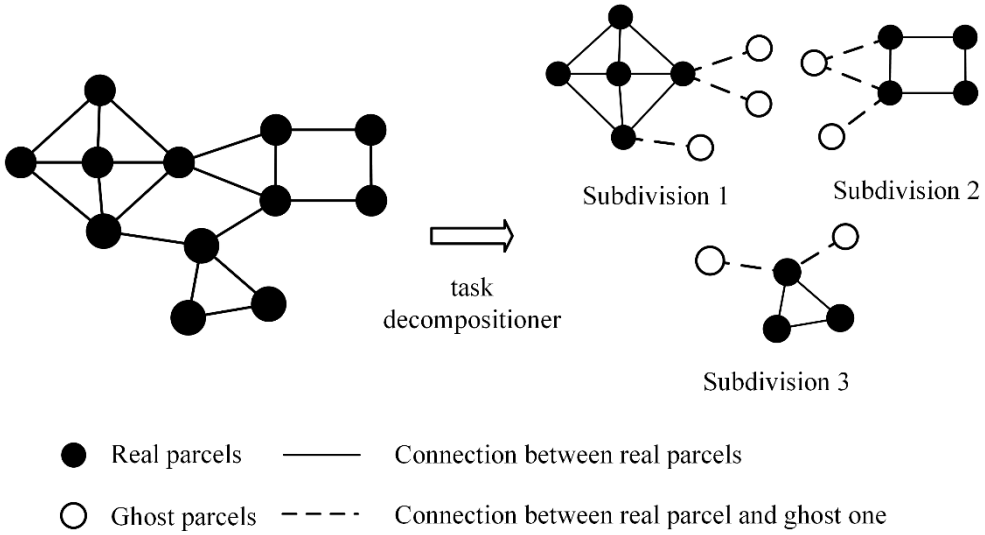


Figure 9. The ghost agent strategy used in the parallelized VCA model.

The simulation was implemented in a parallel simulation framework, 4D-SAS [25]. Its GIS-enabled functionalities provided essential support for geometric computing required by the VCA test model. The 4D-SAS framework accommodated the proposed KCP task decomposition method and the ghost-parcel strategy to reduce the communication overhead.

5. Case study and Experiments

In this paper, two groups of experiments were carried out with the designed parallel VCA urban growth model to evaluate the proposed KCP algorithm. The evaluations compared the proposed KCP method with two other task decomposition methods, GRID, and BSP. The first group used different numbers of divisions, i.e. 4, 8, 16, and 32, to test scalability. The second group was configured with different buffer distances, i.e. 120 m, 240 m, and 360 m, to inspect communication overhead sensitivity. The 4D-SAS simulation framework was deployed on a high-performance computing cluster, consisting of five machines configured as follows; one was used as a master node, three machines were used as simulation engines, and the last one was used for input/output dataset storage. All the servers were directly connected by a dedicated 1 Gbps Ethernet. The hardware and OS of these servers are listed in Table 2.

Table 2. Detailed configuration of the cluster servers.

Type	Master Node	Simulation Engines	Storage Server
CPU	Intel Xeon E5-2620 (2×6 cores, 2.00 GHz each core)	Intel Xeon E5-2620 (2×6 cores, 2.00 GHz each core)	Intel Xeon E5-2620 (2×6 cores, 2.00 GHz each core)
Memory	32 GB (1333 MHz)	32 GB (1333 MHz)	32 GB (1333 MHz)
Hard disk	500 GB (15000 rpm, SAS)	500 GB (15000 rpm, SAS)	1 TB (15000 rpm, SAS)
Kernel	3.10.0-123.el7.x86_64	3.10.0-123.el7.x86_64	
OS	CentOS 7	CentOS 7	Windows Server 2008

5.1. Experimental Results

5.1.1. The Total Simulation Time

In order to evaluate the overall parallel efficiency of KCP, GRID, and BSP, the total simulation time of a 24-year simulation (24 iterations, each iteration represents a year) were recorded as in Table 3.

Table 3. Total computing time for 24 iterations.

Buffer Size	Method	1	4	8	16	32
120 m	KCP	3629.20	792.75	562.85	345.15	179.23
	GRID		997.38	700.09	476.64	294.74
	BSP		854.64	591.54	368.69	249.14
240 m	KCP	3754.20	863.74	598.48	422.56	240.86
	GRID		1037.03	873.04	530.21	352.75
	BSP		985.19	683.28	470.37	347.00
360 m	KCP	3887.63	1036.25	739.14	531.76	323.48
	GRID		1163.13	897.52	611.01	453.94
	BSP		1143.19	849.97	639.43	476.42

As seen in Table 3, KCP achieves the best parallel efficiency with the least simulation time; about 15% less than BSP and 25% less than GRID. The total simulation time for KCP was reduced from over one hour to about three minutes, using 32 cores with a 120 m neighborhood buffer. The simulation results after 24 iterations are shown in Figure 10. Compared to Figure 9, a large part of the parcels has become developed. Parcels near the initial urban area and with good transportation are more likely to be developed. Those areas with limitations, e.g. a protected zone or steep slope, were less likely to be developed.

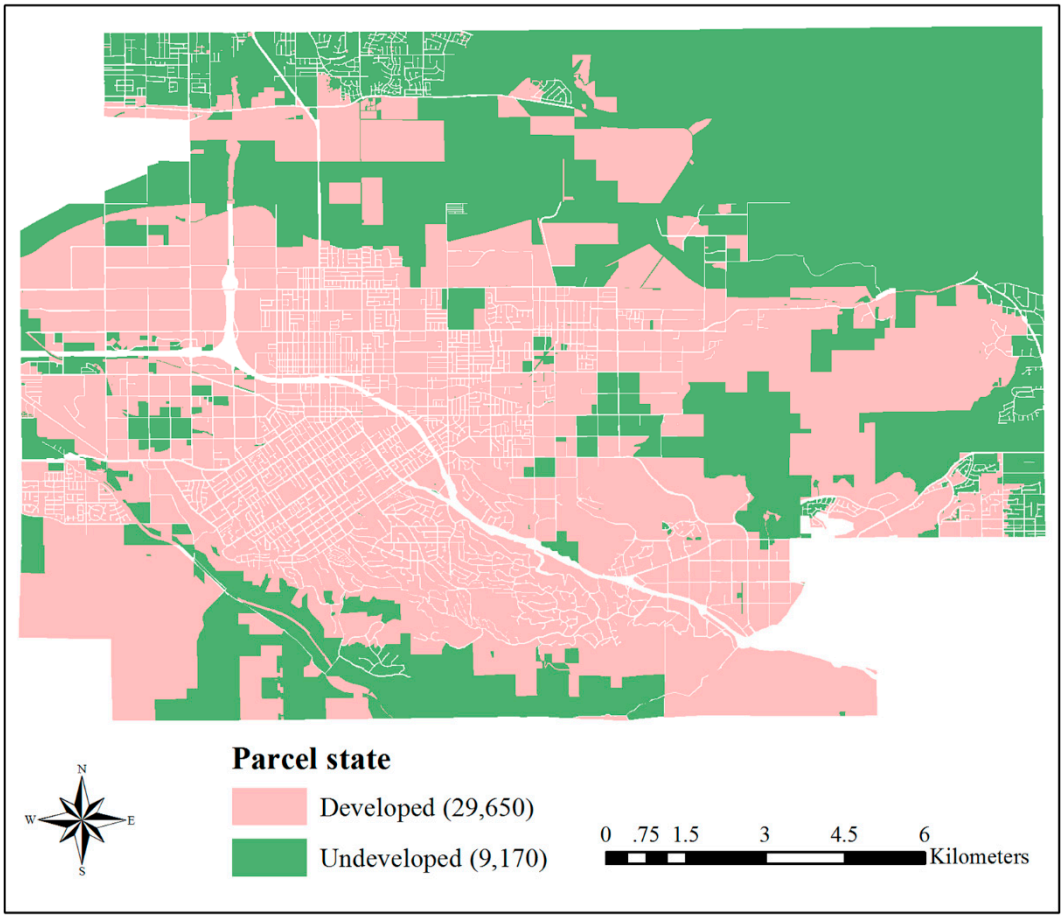


Figure 10. Simulation results of the designed urban growth model.

5.1.2. Execution Time in a Single Iteration

In order to understand how the computation workload and communication overhead affect parallel efficiency, the time of local computing and synchronization were recorded separately with different subdivisions and buffer sizes. The results were recorded as shown in Tables 4 and 5.

Table 4. Average local computing time for an iteration.

Buffer Size	Method	4	8	16	32
120 m	KCP	39.03	27.48	16.47	8.18
	GRID	49.46	34.46	23.20	13.92
	BSP	41.72	28.44	17.12	10.93
240 m	KCP	41.92	29.00	19.06	9.65
	GRID	50.90	37.45	24.75	15.29
	BSP	46.95	31.54	20.58	13.95
360 m	KCP	47.93	33.26	22.71	12.19
	GRID	55.99	42.39	27.83	18.70
	BSP	52.04	36.79	25.71	17.09

Table 5. Average synchronization time for an iteration.

Buffer Size	Method	4	8	16	32
120 m	KCP	0.62	0.70	0.80	0.82
	GRID	0.36	0.54	0.63	0.79
	BSP	1.01	1.13	1.32	1.52
240 m	KCP	1.27	1.51	2.06	2.43
	GRID	0.95	1.32	1.74	2.39
	BSP	2.31	2.62	2.93	3.34
360 m	KCP	3.96	3.71	3.89	4.01
	GRID	2.02	2.34	2.58	3.89
	BSP	5.08	5.66	6.23	6.80

As the buffer size increased, both the local computing time and the synchronization time for all methods increased, which means the computing of each parcel increases as more neighbors become involved. As compared to GRID and BSP, the local computing time of KCP was shorter because both the number of tasks and the computing complexity were considered, based on the centroid distance in KCP. Considering synchronization, KCP takes less time than BSP and more time than GRID, as the communication overhead was lower than that of BSP and higher than that of GRID in terms of the total number of ghost agents.

5.2. Analysis and Discussion

5.2.1. Local Workload Comparison of Different Partition Methods

We assumed that large-sized parcels are surrounded by more neighboring parcels, involving more geometric computing. In this paper, the number of allocated parcels and their total area were examined to analyze workloads obtained from GRID, BSP, and KCP.

(1) The payload distribution of subdivision parcels

Firstly, the normalized number of parcels nn_i in each subdivision was calculated, and the corresponding normalized standard deviation of the number of parcels (NSD-PN) θ was computed according to Equation (7),

$$nn_i = \frac{N_i}{N}$$
$$\theta = \sqrt{\frac{\sum_{i=1}^{k}\left(nn_i - \frac{1}{k}\right)^2}{k}} \quad (7)$$

where K is the subdivision size, i.e. 4, 8, 16, or 32, and N represents the total number of tasks. N_i is the allocated number of tasks for subdivision i. Shown in Table 6 are the results from the BSP method that generates an equal number of parcels for each subdivision. Compared with GRID, KCP results in a better distribution of parcels. As the subdivision number increased, the NSD-PN θ value of GRID and KCP both decreased.

Table 6. Normalized standard deviation of the number of parcels (NSD-PN) of different patitioning methods.

Method	4	8	16	32
KCP	0.086	0.062	0.032	0.015
GRID	0.132	0.086	0.059	0.032
BSP	0	0	0	0

(2) The subdivision parcel area distribution

The same as the NSD-PN calculation, the first step is to compute normalized allocated area ns_j of each subdivision, and then the normalized standard deviation of the resulting partition areas (NSD-PA) μ is computed according to Equation (8),

$$ns_j = \frac{\sum_{i \in sub(j)} s_i}{\sum_{n=1}^{N} s_n}$$
$$\mu = \sqrt{\frac{\sum_{j=1}^{k}(ns_j - \frac{1}{k})^2}{k}} \quad (8)$$

where s_i represents the area of parcel i, and ns_j represents the ration of allocated parcels area for subdivision j. Shown in Table 7, the GRID method resulted in a well-balanced distribution of task areas among subdivisions, while the BSP method resulted in an unbalanced workload. In addition, the μ value decreased as the subdivision sizes increased.

Table 7. Normalized standard deviation of the resulting partition areas (NSD-PA) of different patitioning methods.

Method	4	8	16	32
KCP	0.045	0.041	0.029	0.018
GRID	0.029	0.023	0.016	0.009
BSP	0.104	0.091	0.060	0.038

As shown in Tables 6 and 7, GRID generated the highest NSD-PA and the lowest NSD-PN, while the results produced by BSP were the reverse of the GRID task allocation results. The proposals by KCP were moderate in both aspects. The GRID method weights more spatial task complexity over number of spatial tasks. In contrast, the BSP method focuses on balancing the

number of spatial tasks allocated among subdivisions. The KCP method makes a compromise between these two factors, which can achieve the optimum local workload balance.

5.2.2. Global Communication Overhead Comparison of Partition Methods

As described in Section 4.2, the status synchronization of ghost parcels is the major source of the communication overhead. The number of ghost parcels was used to measure the communication overhead among the three task decomposition methods.

Shown in Table 8, KCP leads to less ghost parcels than BSP but more than GRID. As the subdivision size increased, the ghost parcel number of KCP increased more slowly than that of GRID and BSP. In addition, for the same number of subdivisions, as the buffer size increased, KCP also increased more slowly than GRID and BSP. Therefore, KCP is less sensitive to subdivision and buffer size, which makes it suitable for exploring the effect of neighborhood factors.

Table 8. Global communication overhead in the total ghost agent number.

AOI	Method	4	8	16	32
120 m	KCP	531	682	968	1756
	GRID	190	451	855	1433
	BSP	563	1151	2078	3096
240 m	KCP	1107	1612	2577	4478
	GRID	494	1100	2215	3900
	BSP	1542	2856	4907	7416
360 m	KCP	2022	3446	5029	9290
	GRID	1358	2705	4793	8495
	BSP	2635	5315	8864	13503

As shown by the experiment results, the KCP method achieved the most efficient parallel performance in comparison to GRID and BSP. However, due to the inherent empirical success of *k*-means, the KCP method cannot guarantee optimal task decomposition. Moreover, there is no control on the weighting between the local workload balance and the communication overhead, which is critical for the parallelization efficiency of a VCA model operating on a heterogeneous computing cluster.

6. Conclusions

Compared to raster-based CA models, task decomposition in VCA models is more complex because of heterogeneous parcel distributions and task computing complexity. Existing task decomposition methods employ space partitioning directly (e.g. GRID and BSP) to decompose simulation tasks and thus cannot guarantee a balanced workload and minimum communication. In this paper, a novel task decomposition method based on *k*-means clustering is proposed to partition tasks in VCA models. The KCP method adopts the centroid distance of parcel polygons as a measurement of proximity, which enables both the task numbers and their computing complexity to be taken into consideration. Illustrated by our experimental results, KCP obtains the optimum workload balance with an acceptable communication overhead, therefore achieving high parallel efficiency. KCP can be applied as an effective approach to do task decomposition in parallelization of large-scale detailed VCA, where the variety in computing units cannot be ignored. In the future, we will conduct a detailed study of KCP performance in urban growth models based on VCA involving geometric change. We will add weights to the local workload balance and communication overhead to make task decomposition more responsive to the characteristics of heterogeneous computing clusters.

Acknowledgments: This work is supported by the Natural Science Foundation of China (Grant No.: 41301411), the Natural Science Foundation of Hubei Province (Grant No.: 2015CFB399) and Grand Special of High

resolution On Earth Observation: Application demonstration system of high resolution remote sensing and transportation (Grant NO: 07-Y30B10-9001-14/16).

Author Contributions: Zhenqiang Li and Xuefeng Guan conceived and designed the experiments; Zhenqiang Li performed the experiments; all authors analyzed the data and experimental results; Huayi Wu and Jianya Gong contributed the high-performance computing infrastructure and gave other financial aid; and Zhenqiang Li and Xuefeng Guan wrote the paper. In addition, we sincerely thank Mr. Steve Mcclure for the language polishing and revising.

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

Abbreviations

The following abbreviations are used in this manuscript:

KCP	K-means clustering based task partitioning
KD-Tree	*k*-dimensional tree
GIS	Geographical Information System
OS	Operating System
VCA	Vector-based CA model
KCP	*k*-means clustering partitioning strategy
GRID	Grid partitioning
BSP	Binary space partitioning
NSD-PA	Normalized standard deviation of the area of parcels
NSD-PN	Normalized standard deviation of the number of parcels

References

1. Menard, A. Vecgca: A vector-based geographic cellular automata model allowing geometric transformations of objects. *Environ. Plan. B Plan. Des.* **2008**, *35*, 647–665.
2. Moreno, N.; Wang, F.; Marceau, D.J. Implementation of a dynamic neighborhood in a land-use vector-based cellular automata model. *Comput. Environ. Urban Syst.* **2009**, *33*, 44–54.
3. Moore, K. Resel filtering to aid visualisation within an exploratory data analysis system. *J. Geogr. Syst.* **2000**, *2*, 375–398.
4. Shi, W.; Pang, M.Y.C. Development of voronoi-based cellular automata-an integrated dynamic model for geographical information systems. *Int. J. Geogr. Inf. Sci.* **2000**, *14*, 455–474.
5. Shiyuan, H.; Deren, L. Vector cellular automata based geographical entity. In Proceedings of the 12th International Conference on Geoinformatics, Gävle, Sweden, 7–9 June 2004.
6. Allen, J.; Lu, K. Modeling and prediction of future urban growth in the Charleston region of South Carolina: A GIS-based integrated approach. *Conserv. Ecol.* **2003**, doi:10.5751/es-00595-080202.
7. Abolhasani, S.; Taleai, M.; Karimi, M.; Rezaee Node, A. Simulating urban growth under planning policies through parcel-based cellular automata (parca) model. *Int. J. Geogr. Inf. Sci.* **2016**, *30*, 1–26.
8. González, P.B.; Gómez-Delgado, M.; Benavente, F.-A. Vector-based cellular automata: Exploring new methods of urban growth simulation with cadastral parcels and graph theory. In Proceedings of the CUPUM 2015, Cambridge, MA, USA, 7–10 July 2015.
9. Dahal, K.R.; Chow, T.E. Characterization of neighborhood sensitivity of an irregular cellular automata model of urban growth. *Int. J. Geogr. Inf. Sci.* **2015**, *29*, 475–497.
10. White, R.; Engelen, G. High-resolution integrated modelling of the spatial dynamics of urban and regional systems. *Comput. Environ. Urban Syst.* **2000**, *24*, 383–400.
11. Barreira-González, P.; Gómez-Delgado, M.; Aguilera-Benavente, F. From raster to vector cellular automata models: A new approach to simulate urban growth with the help of graph theory. *Comput. Environ. Urban Syst.* **2015**, *54*, 119–131.
12. Wang, D.; Berry, M.W.; Carr, E.A.; Gross, L.J. A parallel fish landscape model for ecosystem modeling. *Simulation* **2006**, *82*, 451–465.
13. Li, X.; Zhang, X.; Yeh, A.; Liu, X. Parallel cellular automata for large-scale urban simulation using load-balancing techniques. *Int. J. Geogr. Inf. Sci.* **2010**, *24*, 803–820.
14. Kim, I.-H.; Tsou, M.-H.; Feng, C.-C. Design and implementation strategy of a parallel agent-based schelling model. *Comput. Environ. Urban Syst.* **2015**, *49*, 30–41.

15. Guan, Q.; Clarke, K.C. A general-purpose parallel raster processing programming library test application using a geographic cellular automata model. *Int. J. Geogr. Inf. Sci.* **2010**, *24*, 695–722.
16. Jain, A.K. Data clustering: 50 years beyond k-means. *Pattern Recognit. Lett.* **2010**, *31*, 651–666.
17. Dunn, J.C. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *J. Cybern.* **1973**, *3*, 32–57.
18. Cannon, R.L.; Dave, J.V.; Bezdek, J.C. Efficient implementation of the fuzzy c-means clustering algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.* **1986**, *8*, 248–255.
19. Steinbach, M.; Karypis, G.; Kumar, V. A Comparison of Document Clustering Techniques. In Proceedings of the KDD Workshop on Text Mining, Boston, MA, USA, 20 August 2000.
20. Pelleg, D.; Moore, A.W. X-means: Extending k-means with efficient estimation of the number of clusters. In Proceedings of the Seventeenth International Conference on Machine Learning, Stanford, CA, USA, 29 June–2 July 2000.
21. Johnston, K.M. *Agent Analyst: Agent-Based Modeling in Arcgis*; Esri Press: Redlands, CA, USA, 2013; pp. 240–308.
22. Valiant, L.G. A bridging model for parallel computation. *Commun. ACM* **1990**, *33*, 103–111.
23. Rao, D.M. Efficient parallel simulation of spatially-explicit agent-based epidemiological models. *J. Parallel Distrib. Comput.* **2016**, *93*, 102–119.
24. Collier, N.; North, M. Parallel agent-based simulation with repast for high performance computing. *Simulation* **2013**, *89*, 1215–1235.
25. Li, Z.; Guan, X.; Li, R.; Wu, H. 4d-sas: A distributed dynamic-data driven simulation and analysis system for massive spatial agent-based modeling. *ISPRS Int. J. Geo-Inf.* **2016**, doi:10.3390/ijgi5040042.