A Complete Review about Terrain Splitting Optimization

Le Hoang Son
Center for High Performance Computing
Vietnam National University, University of Science
334 Nguyen Trai, Thanh Xuan, Ha Noi, Viet Nam

***Abstract*— Recently, these have been a great interest in the three-dimensional WebGIS system and its applications in various branches such as simulation, modeling, network infrastructure optimization, etc. A challenge task in this kind of system is enhancing the speed of displaying large terrains over the Internet environment. Traditional approaches for this problem were shown to have some limitations, both in splitting methods and in the memory space; thus giving low performance of the three-dimensional WebGIS system. So far, some terrain splitting optimization algorithms have been presented to solve these difficulties, such as Particle Swarm Optimization (PSO-TSA), Genetic Algorithm (GA-TSA), Bread Distributing Task (BDT-TSA) and Stochastic Simulation Test (SIT-TSA). These methods were verified intensively, and showed the advantages over the traditional methods in terms of computational time and saving threshold. However, a systematic comparison between these methods has not been found yet. In this paper, we will give an overview of these kinds of methods, and perform an extensive comparison between them by numerous simulations. Some comments and characteristics of methods are also highlighted.**

Keywords: 3D WebGIS, Heuristic Algorithms, Stochastic Methods, Terrain Splitting Optimization.

I. INTRODUCTION

The three-dimensional WebGIS system has been being considered as a useful tool to model, visualize and display all objects in the real world. Supported by a geographical coordinate system, it provides a flexible way to keep track of an object in a 3D terrain as well as perform some spatial analysis operations such as measurement, visibility, etc. This kind of system has been applied to various branches from Real Estate Information System [9], [11], [28], Tourism [10], [20], [26], [29], Traveling [19], Earthquake Disaster Prevention and Mitigation [25] to many other ones [2], [24], [27].

A challenge task in this kind of system that prevents it to deploy further in practical situations is the capability to handle large *Digital Elevation Model* (DEM) terrains [6], [18]. Normally, it takes some minutes to display a medium terrain. This number is increasing significantly when large terrains are processed. This may lead to the overload of the whole WebGIS system. In some commercial areas, such a slow, unstable 3D WebGIS can cause a great loss of money.

The traditional method for this problem is using *Rendering Engine* [5] including Load, 3D Rendering and Transform steps. However, the limitation of this method can be recognized as the requirement of downloading the whole DEM terrain before processing [12]. Therefore, the larger the DEM terrain is, the longer the waiting time requests. For this reason, it is not used much when focusing on the speed of displaying. Later, some recent methods, presented by Google corporation – *O3D* [23] and the authors in [14] - *JSG* have elicited a new approach to deal with this problem. Instead of downloading the whole DEM terrain, they divide it into some pieces, and these small ones are transferred to clients one after another. Then, each part is constructed to display (3D Rendering) using the *Painter's algorithm* [5] or more advanced *Z-buffering* [1]. While rendering a part, some other parts are transferred to clients, and the rendering step is continued until all parts are totally sent. Indeed, this process makes us feel that the 3D scene is displayed immediately.

Nevertheless, these approaches have some limitations about the spatial characteristic between regions and the dividing time. Indeed, a new result from [12] namely SESA was presented. This algorithm used a pre-processing step based on geometric processing between polygons to arrange some elements into specific blocks. Thus, results can be quickly found from those blocks.

However, these are still two weaknesses of this algorithm. *Firstly*, the "suitable" solution found by SESA is not optimal in many cases. In the other words, the saving threshold is not the smallest one. *Secondly*, SESA spends long time on finding solution. After each number of partitions, the generator re-adjusts the parameters if it cannot find any possible solution. This makes the answer time is really long in case of no satisfied partition.

Recently, some terrain splitting optimization algorithms have been presented to solve these difficulties, such as Bread Distributing Task (BDT-TSA) [13], Stochastic Simulation Test (SIT-TSA) [17], Particle Swarm Optimization (PSO-TSA), Genetic Algorithm (GA-TSA) [22]. These methods were verified intensively, and showed the advantages over the traditional methods in terms of the computational time and the saving threshold.

In this paper, we will make an extensive analysis of these

This work was supported by Vietnam National University, Hanoi [QGTĐ.11.01].

methods and perform a complete comparison between them in two aspects: the saving threshold and the computational time on the benchmark DEM terrain dataset from the Bolzano-Bolzen province, Italy [14].

The rests of the paper are organized as follows. Section 2 introduces the optimization problem. Reviews of some terrain splitting optimization algorithms will be presented in Section 3. In the two next sections, the comparisons about saving threshold and computational time will be shown, respectively. Section 6 will summarize the findings from two previous sections and generate some fuzzy rules from them. Finally, we will make conclusions and future works in the last section.

II. The Optimization Problem

Our problem can be understood as follows: We have to split a 3D DEM terrain following by some polygons in *2D Polygonal Vector Data* (2PVD) and the number of processors k in the system.

$$J_1 = \sum_{i=1}^{k} SP_i \to \min \quad , (1)$$

Where SP_i , $i = \overline{1,k}$ are the areas of the small DEM terrains in processor i. Some constraints are:

$$\begin{cases} \left|SP_i\right| \le \alpha \times S_{DEM} \\ \left|SP_i - SP_j\right| \le \varepsilon \times S_{DEM} \\ i = \overline{1,k};\ j = \overline{1,k}; i \ne j \end{cases} \quad . (2)$$

S_{DEM} is the area of original 3D DEM terrain. ε is a given error. The parameter α is called the *saving threshold* of memory space in each processor. Being mentioned in the previous section, it should be as minimal as possible to reduce the memory space in each processor.

III. Some Terrain Splitting Optimization Algorithms

Nguyen et al. (2011) [22] have shown two algorithms to solve the original problem. *The first one* based on Genetic Algorithm [4], [7] namely *GA-TSA* employs some ideas of natural evolution, such as inheritance, mutation, selection, and crossover for finding the best saving threshold in a search area. In this algorithm, an individual is a collection of indexes of all polygons in 2PVD that represent for all blocks in a current solution. Then, through a fitness function, all individuals are sorted in the ascending order, and half of them are selected to reproduce a new generation by the mean of Cross Over and Mutation operations. After pre-defined maximal iteration steps, the best generation is found, and the saving threshold can be extracted from it. However, in some cases, the last solution sometimes does not give a possible saving threshold due to bad initialization. Besides, the difference of areas between blocks in the last solution may be larger than ε. For these reasons, some modifications should be performed, either selecting another initial population or increasing parameter ε . Anyway, this algorithm still guarantees obtaining better results than SESA does.

The second algorithm in the literature [22] started with an idea of Swarm Optimization, which is considered to be the most suitable strategy among all of Heuristic Optimization. The chosen algorithm to develop is Particle Swarm Optimization (PSO) which was invented by Dr. Kennedy in 1995 [8]. Indeed, the algorithm was named *PSO-TSA*. The basic idea of this algorithm lies on the *Seed Procedure*. Basically, k seeds are evenly distributed in the space. Each seed represents for a number of polygons in 2PVD. Then, the authors calculate and check the constraints from A_1 to A_3. If these criteria are not met, PSO algorithm is used to generate a new population until the stopping condition is reached. In the last iteration, the particle holding *gBest* value will be outputted if it satisfies the constraints. Nevertheless, similar to GA-TSA, it still remains the same disadvantages. Moreover, both algorithms do not guarantee success in all cases.

The authors in [13] have presented another solution for the considered problem. They employed some basic principles of Particle Swarm Optimization and Bread-Distributing scenario. In essence, the new algorithm is a modification of the PSO-TSA because it uses all phases of Particle Swarm Optimization from swarm initialization, searching for local and global best particles to calculating new solutions from the previous one. The only change in comparison with PSO-TSA is the using of an idea: *Bread Distributing Task*. Indeed, the algorithm was named as *BDT-TSA*. Initially, all polygons in 2PVD are assigned to k blocks with k is the number of processors in the system by the mean of R - SESA algorithm [15], [16]. Then, the fitness's value of each block is calculated; thus updates two best values: pbest and gbest. This process is similar to the main PSO algorithm. However, in the next step, only the "worst" and the "best" block are changed and affected together. This means that a polygon in the "worst" block will be transferred to the "best" one. This polygon is randomly chosen through pbest and gbest values. The algorithm repeats this process until the maximal iteration step is reached.

These are some points that make BDT-TSA be different with PSO-TSA. *Firstly*, a particle is not a solution. Instead, a solution is a collection of particles. *Secondly*, the velocities and positions of particles are not defined. The only thing that makes a solution change is the index of element in the "worst" particle, which is calculated through pbest and gbest values. In the experiments, BDT-TSA was shown to be better than SESA in both the saving threshold and the computational time.

The last optimization technique for the problem (1) – (2) has been found from the literature [17]. In that paper, the authors proposed a new algorithm named *Stochastic sImulation Test based Terrain Splitting Algorithm* (*SIT-TSA*). In essence, it is a stochastic and agent - based approach that acts following by a basic principle: "*It is supposed to be no satisfied solution with probability* $1-p$ *after a series of*

failed stochastic simulation tests on various possibilities derived from the original sample". In the other word, this principle behaves as a similar way to Monte Carlo method - a class of computational algorithms that rely on repeated random sampling to compute their results [3], [21]. The SIT-TSA acts like the scenario above. An agent is randomly initiated at a polygon. By using a local search method, an ordered list of polygons in 2PVD is established. Then, by multiple tests with random "sticks" dividing this list into k blocks, a candidate list of this agent is set up. This process is repeatedly performed for other agents. Finally, the optimal solution with minimal saving threshold parameters will be chosen from all the candidate lists. In addition, parallel computing is used as an important aid for the reduction of total computing time due to independent works between agents.

The SIT-TSA method converges to the global solution instead of local one due to the best solution selection process among all agents. Moreover, it can answer quickly whether a solution may exist for a given parameters ε or not. This method was shown to obtain more successful results than PSO-TSA.

All these optimization algorithms were shown to be suitable for the considered problem. Moreover, they outperform the traditional methods. However, there is not a best method among them. Each method has own advantages and situations. Thus, our mission is to find which algorithm we should choose for a specific input.

In what follows, we will make the comparisons between them in two aspects: the saving threshold and the computational time.

IV. The Comparison Of Saving Threshold

A. The statistics by the number of polygons

In the experiment, we have run four algorithms above following by the number of polygons and the number of processors in two kinds of DEM terrain: medium (sizes: 4039 x 6529 ~ 24 million points) and large (sizes: 8024 x 9621 ~ 77 million points). The outputs are the saving thresholds and the computational times.

We compare the saving thresholds of four algorithms for a specific number of polygons and processors, and find the smallest one. Then, we increase the number of cases for the algorithm that has the smallest value of saving threshold by one. The statistics are grouped following by the number of polygons. Results are illustrated in Fig. 1.

From this figure, we may see that SIT-TSA obtains better results than other algorithms in most cases. For example, when the number of polygons is 5000, the numbers of cases generated by GA-TSA, PSO-TSA, BDT-TSA and SIT-TSA are 1, 3, 5 and 7, respectively. If we consider the total number of cases, for a given number of polygon is 100 percents, the percent values of all algorithms in the example above will be 6.25%, 18.75%, 31.25% and 43.75%. This means that if the number of polygons in the input data is 5000, and the SIT-TSA algorithm is chosen, the possibility for the saving threshold of this algorithm to be the smallest one is nearly 44%.

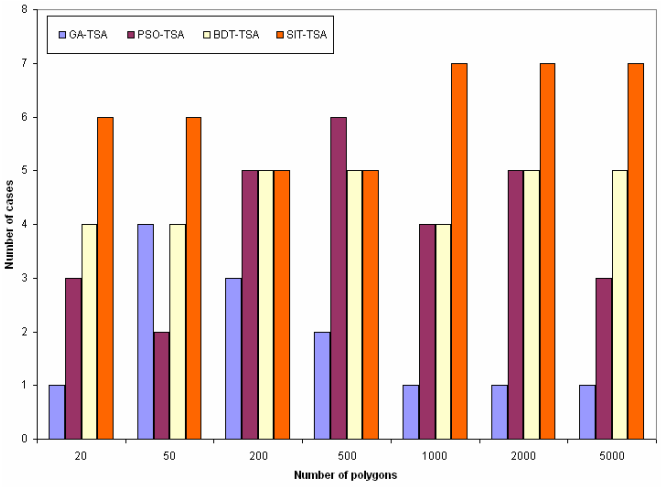

Figure 1. The statistics of saving threshold by the number of polygons

The percent values of SIT-TSA in cases of 20, 50, 200, 500, 1000 and 2000 polygons are 42.85%, 37.5%, 27.78%, 27.78%, 43.75% and 38.89%, respectively. Thus, we should choose the SIT-TSA algorithm if the number of polygons is large (over 1000) or small (below 50). In case the number of polygons is medium, PSO-TSA should be chosen because it contributes the largest percent values among all, i.e. 33.34% (500 polygons).

Through this figure, we can recognize another result that BDT-TSA is better than PSO-TSA and GA-TSA. Except 500 polygons, the number of cases generated by BDT-TSA is higher than the ones of PSO-TSA. Similarly, except 50 polygons, PSO-TSA creates more cases than GA-TSA does. Therefore, we have an order of algorithms to be chosen, that is SIT-TSA, BDT-TSA, PSO-TSA and GA-TSA. This order helps us to choose the substitute for the main algorithm mentioned above. For example, when the number of polygons is large or small, if we do not choose SIT-TSA, BDT-TSA is selected as the substitute. When the number of polygons is medium, SIT-TSA is chosen as the substitute for the PSO-TSA.

B. The statistics by the number of processors

This subsection performs the same statistics with the previous one, but following by the number of processors. Results are depicted in Fig. 2.

This figure shows that we should choose the SIT-TSA algorithm if the numbers of processors are medium and small (from 2 to 7). The numbers of cases for SIT-TSA with 2, 3, 4 processors are 10, 11, 8, respectively. The equivalent percent values are 40%, 45.8% and 47.1%. In this situation, the order of algorithms to be chosen is SIT-TSA, BDT-TSA, PSO-TSA and GA-TSA.

When the number of processors is large (from 8 to 16), BDT - TSA should be chosen as it contributes more cases than other algorithms. For example, when we use 16 processors, the

numbers of cases generated by GA-TSA, PSO-TSA, BDT-TSA and SIT-TSA are 2, 1, 10 and 6, respectively. The equivalent percent values are 10.53%, 5.26%, 52.63% and 31.58%. The order of algorithms is BDT-TSA, PSO-TSA, SIT-TSA and GA-TSA.

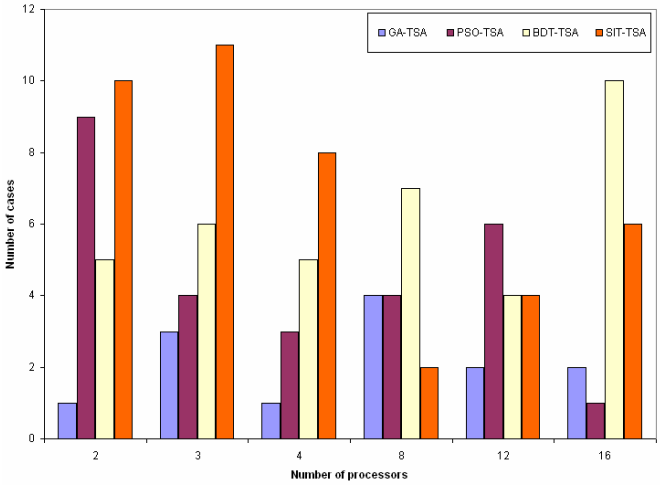

Figure 2. The statistics of saving threshold by the number of processors

C. *Average saving threshold by number of polygons*

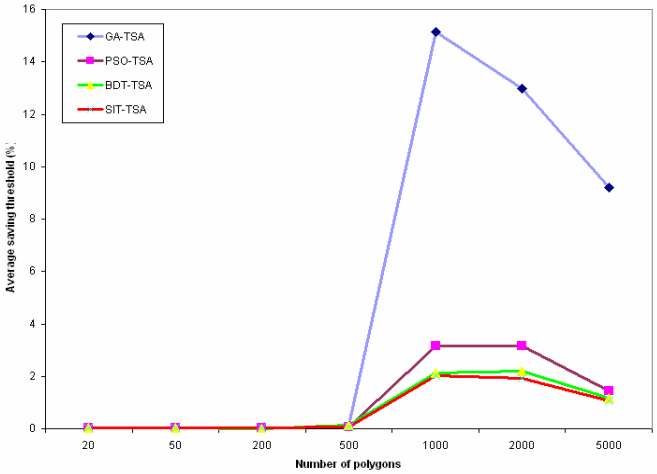

Figure 3. The average saving thresholds on the medium DEM terrain

In Fig. 3, we compare the average saving thresholds of four algorithms following by the number of polygons. These values are calculated on the medium DEM terrain. Fig. 3 shows that there is little difference between four algorithms when the number of polygons is below 500. The maximal differences between the algorithms in cases of 20, 50, 200 and 500 polygons are 0.005, 0.004, 0.026 and 0.099, respectively. However, when the number of polygons is large, the difference is getting obvious. The maximal differences between the algorithms in cases of 1000, 2000 and 5000 polygons are 13.1, 11.04 and 8.16, respectively.

The SIT-TSA algorithm obtains the smallest average saving thresholds among all other ones. For example, when the number of polygons is 1000, the threshold values of GA-TSA, PSO-TSA, BDT-TSA and SIT-TSA are 15.1, 3.14, 2.13 and 2.04, respectively. From 500 polygons afterward, we can clearly see that the line of GA - TSA is higher than the ones of PSO-TSA, BDT-TSA and SIT-TSA. Thus, the order of all algorithms in Fig. 3 should be SIT-TSA, BDT-TSA, PSO-TSA and GA-TSA.

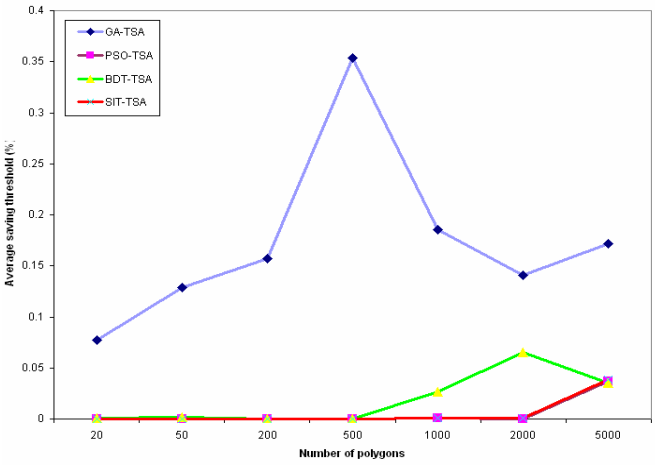

Figure 4. The average saving thresholds on the large DEM terrain

In Fig. 4, we also calculate the average saving thresholds of four algorithms similar to Fig. 3, but on the large DEM terrain. Through this figure, GA-TSA is shown to obtain higher saving threshold values than other algorithms because its line is quite far from the remains. Below 500 polygons, the three left lines are nearly the same. The maximal differences between these lines in cases of 20, 50, 200 and 500 polygons are 0.000923, 0.001276, 0.000079 and 0.000179. For the range [500, 5000] polygons, the line of BDT-TSA is higher than the rests. Thus, BDT-TSA is suitable for the input whose numbers of polygons are medium and small, and the DEM terrain is large.

Although the two lines of PSO-TSA and SIT-TSA are nearly the same for all cases, PSO-TSA still obtains smaller values of saving threshold than SIT-TSA does. For example, the difference between these lines in case of 20 polygons is 0.000008. More polygons are added, more obvious the difference is. When the number of polygons is 5000, the difference is increased to 0.001543. Thus, the order of all algorithms in this figure is PSO-TSA, SIT-TSA, BDT-TSA and GA-TSA.

D. *Average saving threshold by number of processors*

Now, we perform the same calculations as in the previous subsection, but for the number of processors. Results are shown in Fig. 5 and Fig. 6.

In Fig. 5, the line of GA-TSA is higher than other ones. In the range [2, 12] processors, SIT-TSA is better than BDT-TSA, and BDT-TSA is better than PSO-TSA. The threshold value of BDT-TSA is approximately 2.11 times higher than the one of SIT-TSA. These numbers in cases of PSO-TSA and GA-TSA are 6.73 and 48.1, respectively. When the number of processors is 16, the differences between SIT-TSA, BDT-TSA and PSO-TSA are quite small (~ 0.02). Thus, we should choose the number of processors in the range above to get the

distinct results of all algorithms. The order of all algorithms in this situation is SIT-TSA, BDT-TSA, PSO-TSA and GA-TSA.

Fig. 6 shows a different result with the figure above. Except GA-TSA which produces the worst result in comparison with the rests, the other algorithms always create the values of saving threshold below 0.051. In the range [2, 4] processors, the line of BDT-TSA is higher than the ones of PSO-TSA and SIT-TSA. The maximal difference between BDT-TSA and these algorithms is 0.05 when the number of processors is two. The difference is getting smaller when more processors are added. The saving thresholds of three algorithms are approximately equal in the range (4, 8] processors.

The last range (8, 16] shows the saving threshold of BDT-TSA is smaller than the ones of PSO-TSA and SIT-TSA. Thus, we should choose the BDT-TSA if the number of processors is large. When the numbers of processors are medium and small, PSO-TSA and SIT-TSA are selected, respectively.

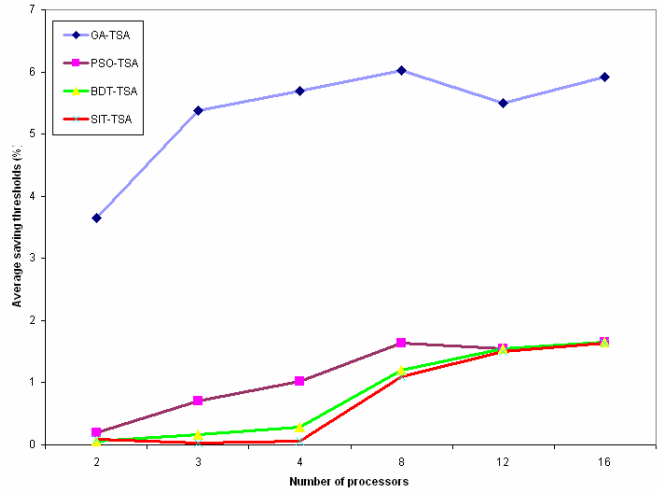

Figure 5. The average saving thresholds on the medium DEM terrain

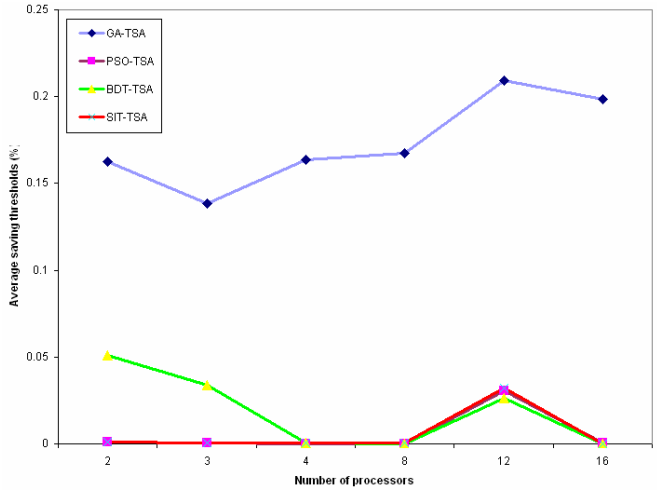

Figure 6. The average saving thresholds on the large DEM terrain

E. Advanced statistics of the saving threshold

In Fig. 7, we summarize the statistics and draw the histograms of all algorithms. Results show that both the minimal and maximal values of SIT-TSA, BDT-TSA are smaller than the ones of PSO-TSA and GA-TSA, respectively. In SIT-TSA, 25 percents of the saving thresholds are smaller than 0.000321. The numbers in cases of BDT-TSA, PSO-TSA and GA-TSA are 0.0003035, 0.000060 and 0.027977, respectively. We can recognize that SIT-TSA is not better than PSO-TSA and BDT-TSA in this situation. However, the median value of SIT-TSA points out that 50 percents of the saving thresholds are smaller than 0.003707. This number is smaller than PSO-TSA (0.008759), BDT-TSA (0.0187035) and GA-TSA (0.163955).

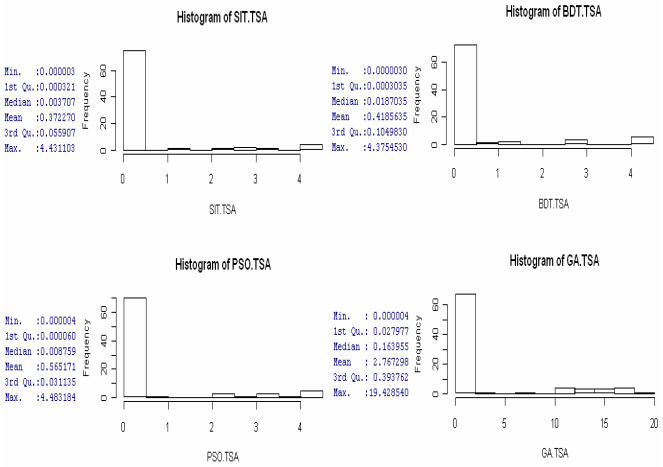

Figure 7. Histograms and summary statistics of all algorithms

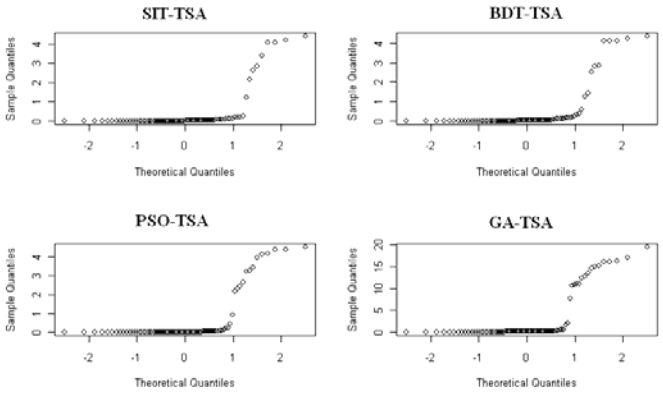

Figure 8. Normal Q-Q Plots of all algorithms

75 percents of saving thresholds state that PSO-TSA is the best method with value 0.031135 in comparison with SIT-TSA (0.055907), BDT-TSA (0.1049830) and GA-TSA (0.393762). However, 25 percents of left saving thresholds of PSO-TSA are larger values than SIT-TSA (max = 4.431103) and BDT-TSA (max = 4.3754530). Indeed, this algorithm is somehow not better than SIT-TSA and BDT-TSA. The mean values clearly show the order of algorithms to be chosen: SIT-TSA, BDT-TSA, PSO-TSA and GA-TSA.

We also draw the normal Q-Q plots of all algorithms. From Fig. 8, the saving threshold values of all algorithms do not follow the standard distribution.

F. The saving threshold prediction

In this subsection, we will construct the models to predict the saving threshold values of all algorithms following by the size of DEM terrain (*DEM*), the number of polygons in the terrain (*Pol*) and the number of processors in the system (*Proc*). Fig. 9 shows the distributions of the saving thresholds of all algorithms. Obviously, the threshold values in the first 43 tests are much higher than the remains. In fact, they are conducted from the medium DEM terrain. Indeed, small DEM terrains often create large values of saving threshold than large DEMs do.

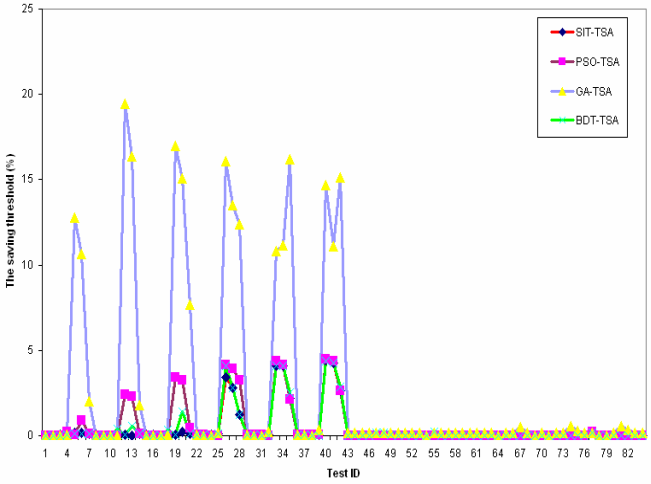

Figure 9. The distributions of the saving thresholds of all algorithms

In what follow, we use the linear regression toolbox in the software package R[1] to construct the prediction models for all algorithms.

$$SIT - TSA = 0.8290003 - 0.0001839 * DEM + 0.0659394 * Proc + 0.0001262 * Pol \quad (3)$$

$$PSO - TSA = 1.6900059 - 0.00028 * DEM + 0.045665 * Proc + 0.000181 * Pol \quad (4)$$

$$GA - TSA = 9.313887 - 0.0013 * DEM + 0.0000002 * Proc + 0.001042 * Pol \quad (5)$$

$$BDT - TSA = 0.9883551 - 0.0002 * DEM + 0.061936 * Proc + 0.000141 * Pol \quad (6)$$

From these models, we can calculate the predictive saving threshold values for all algorithms following by a certain input. Then, we can choose the algorithm that produces the smallest value among all. The experimental results also show that the *Akaike Information Criterion* (AIC) values of SIT-TSA, PSO-TSA, GA-TSA and BDT-TSA are 3.3, 22.25, 259.4 and 9.58, respectively. They reconfirm the order of algorithms to be chosen, that is SIT-TSA, BDT-TSA, PSO-TSA and GA-TSA.

V. The Comparison Of Computational Time

A. Average computational time by the number of polygons

In the previous section, we have considered the saving threshold of all algorithms. In addition to this criterion, the computational time should be examined carefully. An algorithm is not effective if it runs too long; although it can produce the smallest value of saving threshold. Therefore, in this section, we have also performed the same experiments with Section 4, but focused intensively on the computational time.

In Fig. 10, we investigate the average computational time of all algorithms following by the number of polygons on the medium DEM terrain. Results show that SIT-TSA is the most effective algorithm when the numbers of polygons are small and medium (below 200). In these cases, GA-TSA runs approximately 35.4 times longer than SIT-TSA on average. These numbers in PSO-TSA, BDT-TSA are 279 and 539 times, respectively. However, these differences are getting smaller when the number of polygons increases, i.e. 1.03, 14 and 4.14 times longer than SIT-TSA in cases of GA-TSA, PSO-TSA and BDT-TSA, respectively. The order of algorithms in this situation is SIT-TSA, GA-TSA, PSO-TSA and BDT-TSA.

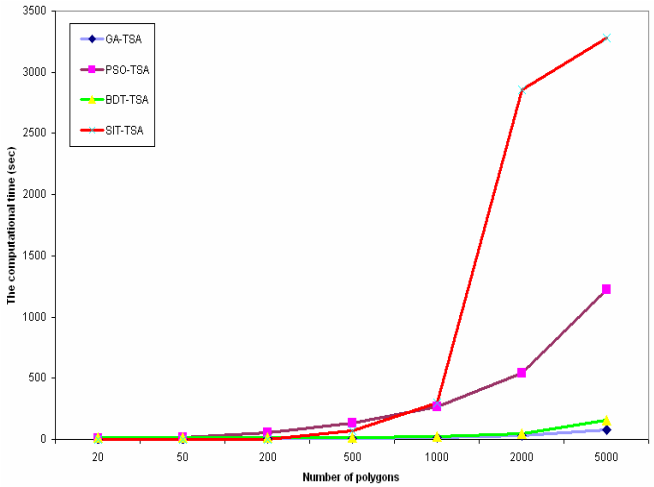

Figure 10. The average computational time on the medium DEM terrain

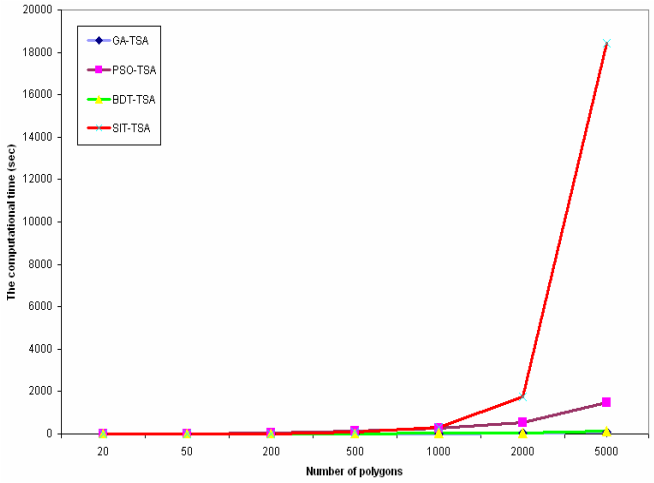

Figure 11. The average computational time on the large DEM terrain

When the number of polygons is large, GA-TSA is shown to be the fastest algorithm among all. The computational times of PSO-TSA, BDT-TSA and SIT-TSA are 15.3, 1.72 and 37.3 times longer than GA-TSA, respectively. When the number of polygons increases, these numbers are larger as the amplitudes

[1] The software package R is available at: http://cran.r-project.org

of lines are expanded. Thus, in this situation, the order of algorithms is GA-TSA, BDT-TSA, PSO-TSA and SIT-TSA.

In GA-TSA, the incremental level between two numbers of polygons is 2.19 times. These numbers in cases of PSO-TSA, BDT-TSA and SIT-TSA are 2.49, 1.65 and 17.2, respectively. Indeed, they explain why the SIT-TSA turns from the fastest algorithm to the slowest one as shown in above.

In Fig. 11, we calculate the average computational time on the large DEM terrain. The results are similar to the ones on the medium DEM. For the small and medium number of polygons, SIT-TSA still predominates over the rests. The computational times of GA-TSA, PSO-TSA and BDT-TSA are 41.9, 367 and 764 times longer than the one of SIT-TSA, respectively. These numbers are longer than the equivalent results in Fig. 10 about 1.18, 1.31 and 1.41 times. Thus, this fact explains why GA-TSA tends to the most stable algorithm among all because it keeps a little change of the computational time when the sizes of DEM terrain are increased.

For the large number of polygons, PSO-TSA, BDT-TSA and SIT-TSA run 16.1, 1.51, 72.1 times longer than GA-TSA does, respectively. The incremental levels between two numbers of polygons in cases of GA-TSA, PSO-TSA, BDT-TSA and SIT-TSA are 2.22, 2.57, 1.56 and 21.5, respectively. The order of algorithms is kept intact as in the same situation in Fig. 10.

B. *Average computational time by the number of processors*

This subsection investigates the average computational times of all algorithms following by the number of processors. Results are depicted in Fig. 12 and Fig. 13.

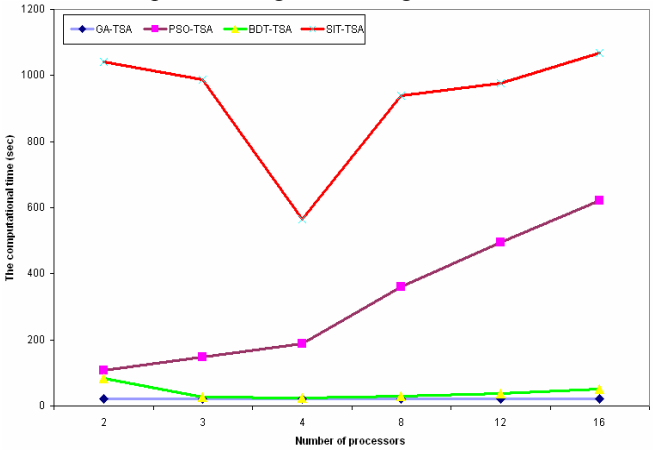

Figure 12. The average computational time on the medium DEM terrain

Fig. 12 shows that GA-TSA is the fastest algorithm in all numbers of polygons. The computational times of PSO-TSA, BDT-TSA and SIT-TSA are 14.7, 1.97 and 43.7 times longer than GA-TSA. The order of algorithms in this situation is GA-TSA, BDT-TSA, PSO-TSA and SIT-TSA.

Fig. 12 also points out that the processing times per processor of GA-TSA, PSO-TSA, BDT-TSA and SIT-TSA are 4.68, 45.9, 11.1 and 209 seconds, respectively. As such, more processors are added, more effective the GA-TSA shows. Besides, these numbers help us to give a predictive value of computational time when processing a certain number of processors.

In Fig. 13, the computational times of PSO-TSA, BDT-TSA and SIT-TSA are 15.9, 1.63 and 137 times longer than GA-TSA. The processing times per processor of GA-TSA, PSO-TSA, BDT-TSA and SIT-TSA are 4.8, 49.3, 8.8 and 962 seconds, respectively. Except SIT-TSA, these numbers are approximately equal to the equivalent results in Fig. 12. When the sizes of the DEM terrain increase, the processing time per processor of SIT-TSA increases drastically. Thus, we should not choose SIT-TSA if the number of processors is large. The order of all algorithms is kept intact in this situation.

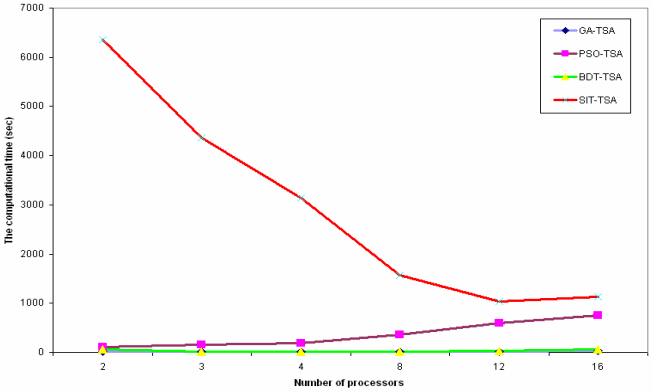

Figure 13. The average computational time on the large DEM terrain

C. *The computational time prediction*

In Fig. 14, we study the distribution of the computational times of all algorithms. From this figure, we can easily recognize that the times are small in the first 43 tests. For the remains, the computational times, especially SIT-TSA, increase remarkably.

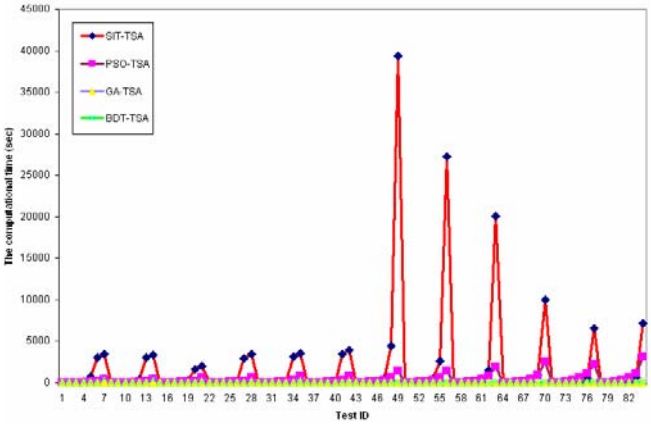

Figure 14. The distributions of the computational time of all algorithms

We also use the linear regression toolbox in the software package R to build the predictive models of computational times for all algorithms.

$$SIT - TSA = -2666.8467 + 0.5031 * DEM - 156.8086 * Proc + 2.1872 * Pol \quad , (7)$$

$$PSO - TSA = -701.16554 + 0.09586 * DEM + \quad , (8)$$

$$GA - TSA = -2.8628398 + 0.000381 * DEM + 0.1578227 * Proc + 0.016741 * Pol \quad (9)$$

$$BDT - TSA = 8.14011 + 0.000001 * DEM + 0.000001 * Proc + 0.02461 * Pol \quad (10)$$

The AIC values of these algorithms are 1409.6, 965.41, 117.1 and 686.7, respectively. They re-confirm that the order of algorithms to be chosen for the sake of computational time is GA-TSA, BDT-TSA, PSO-TSA and SIT-TSA.

VI. Fuzzy Rules Generation

Through the evaluations of the saving threshold and the computational time, we have extracted the fuzzy rules for these criteria. Results are described in Table 1 and Table 2 (Appendix). These rules can help us to determine the suitable algorithm for a given data input. For example, the following rule specifies a list of algorithms that optimize the saving threshold,

If "DEM is large" and "The number of processors is medium" and "The number of polygons is small" then "the order of algorithms to be chosen is PSO - TSA, SIT - TSA, BDT - TSA and GA - TSA" (11)

If we want to focus on the computational time criterion, the rule in (11) is replaced by

If "DEM is large" and "The number of processors is medium" and "The number of polygons is small" then "the order of algorithms to be chosen is SIT - TSA, GA - TSA, PSO - TSA and BDT - TSA" (12)

If we want to calculate the predictive values of these criteria for all algorithms in the list, the equations from (3) to (10) can be used.

VII. Conclusions

This paper aimed to introduce some new results of Terrain Splitting Optimization problem, which is considered as a promising topic in Geographic Information Systems. We presented their mechanisms, advantages as well as limitations. Besides, we also examined the characteristics of four best current methods following by two criteria: the saving threshold and the computational time. The comparisons between them were made completely and summarized through the fuzzy rules. Thus, these highlights could help us to choose the appropriate algorithm for a given data input.

Future works will concern some methods to query information on three-dimensional terrains as well as exploit attribute information.

Appendix

TABLE I. Fuzzy Rules of The Saving Threshold

No.	DEM	Processors	Polygons	The order of algorithms
1	Medium	Small	Small	SIT - TSA, BDT - TSA, PSO - TSA and GA - TSA.
2	Medium	Small	Medium	PSO - TSA, SIT - TSA, BDT - TSA and GA - TSA
3	Medium	Small	Large	SIT - TSA, BDT - TSA, PSO - TSA and GA - TSA.
4	Medium	Medium	Small	SIT - TSA, BDT - TSA, PSO - TSA and GA - TSA.
5	Medium	Medium	Medium	PSO - TSA, SIT - TSA, BDT - TSA and GA - TSA
6	Medium	Medium	Large	SIT - TSA, BDT - TSA, PSO - TSA and GA - TSA.
7	Medium	Large	Small	BDT - TSA, PSO - TSA, SIT - TSA and GA - TSA.
8	Medium	Large	Medium	PSO - TSA, SIT - TSA, BDT - TSA and GA - TSA
9	Medium	Large	Large	BDT - TSA, PSO - TSA, SIT - TSA and GA - TSA.
10	Large	Small	Small	SIT - TSA, BDT - TSA, PSO - TSA and GA - TSA.
11	Large	Small	Medium	SIT - TSA, BDT - TSA, PSO - TSA and GA - TSA.
12	Large	Small	Large	SIT - TSA, BDT - TSA, PSO - TSA and GA - TSA.
13	Large	Medium	Small	PSO - TSA, SIT - TSA, BDT - TSA and GA - TSA
14	Large	Medium	Medium	PSO - TSA, SIT - TSA, BDT - TSA and GA - TSA
15	Large	Medium	Large	PSO - TSA, SIT - TSA, BDT - TSA and GA - TSA
16	Large	Large	Small	BDT - TSA, PSO - TSA, SIT - TSA and GA - TSA
17	Large	Large	Medium	BDT - TSA, PSO - TSA, SIT - TSA and GA - TSA
18	Large	Large	Large	BDT - TSA, PSO - TSA, SIT - TSA and GA - TSA

TABLE II. Fuzzy Rules of The Computational Time

No.	DEM	Processors	Polygons	The order of algorithms
1	Medium	Small	Small	SIT - TSA, GA - TSA, PSO - TSA and BDT - TSA
2	Medium	Small	Medium	SIT - TSA, GA - TSA, PSO - TSA and BDT - TSA
3	Medium	Small	Large	GA - TSA, BDT - TSA, PSO - TSA and SIT - TSA.
4	Medium	Medium	Small	SIT - TSA, GA - TSA, PSO - TSA and BDT - TSA
5	Medium	Medium	Medium	SIT - TSA, GA - TSA, PSO - TSA and BDT - TSA
6	Medium	Medium	Large	GA - TSA, BDT - TSA, PSO - TSA and SIT - TSA.
7	Medium	Large	Small	SIT - TSA, GA - TSA, PSO - TSA and BDT - TSA
8	Medium	Large	Medium	SIT - TSA, GA - TSA, PSO - TSA and BDT - TSA

9	Medium	Large	Large	GA - TSA, BDT - TSA, PSO - TSA and SIT - TSA.
10	Large	Small	Small	SIT - TSA, GA - TSA, PSO - TSA and BDT - TSA
11	Large	Small	Medium	SIT - TSA, GA - TSA, PSO - TSA and BDT - TSA
12	Large	Small	Large	GA - TSA, BDT - TSA, PSO - TSA and SIT - TSA.
13	Large	Medium	Small	SIT - TSA, GA - TSA, PSO - TSA and BDT - TSA
14	Large	Medium	Medium	SIT - TSA, GA - TSA, PSO - TSA and BDT - TSA
15	Large	Medium	Large	GA - TSA, BDT - TSA, PSO - TSA and SIT - TSA.
16	Large	Large	Small	SIT - TSA, GA - TSA, PSO - TSA and BDT - TSA
17	Large	Large	Medium	SIT - TSA, GA - TSA, PSO - TSA and BDT - TSA
18	Large	Large	Large	GA - TSA, BDT - TSA, PSO - TSA and SIT - TSA.

ACKNOWLEDGMENT

The authors are greatly indebted to the Editor-in-Chief Prof. Soumen Ganguly; anonymous reviewers for their comments and suggestions which improved the quality and clarity of paper.

REFERENCES

[1] Catmull E., A subdivision algorithm for computer display of curved surfaces (Dissertation for the Doctoral Degree). USA: University of Utah, 1974.

[2] CUI YongyiZHOU, DongruWAN, GangFU Huasheng, "Web VRGIS Study Based on VRML," Computer Engineering, vol. 8, 2002.

[3] Eckhardt Roger, "Stan Ulam, John von Neumann, and the Monte Carlo method," Los Alamos Science, vol. 15, pp. 131-137, 1987.

[4] Fraser Alex, Donald Burnell, Computer Models in Genetics. New York: McGraw-Hill, 1970.

[5] Foley J, Van D A, Feiner S K, Hughes J F., Computer Graphics: Principles and Practice. Reading, MA, USA: Addison-Wesley, 1990.

[6] GAO Ying-jie et al., "Development of DEM on 1:10000 Scale and Its Application in Geo-sciences," Journal of Anhui Agricultural Sciences, vol. 2, 2009.

[7] Holland J H., "Adaptation in natural and artificial system," Ann Arbor, USA: The University of Michigan Press, 1975.

[8] Kennedy, J. and Eberhart, R. C., "Particle swarm optimization," Proceedings of IEEE International Conference on Neural Networks, Perth, WA, Australia, 27 November - 01 December 1995, pp. 1942-1948.

[9] LI Lin-lin, CAO Kai-bin, GUAN Bin, ZHU Wei-dong, "Study on the Application of WebGIS and VR in the Information System of Real Estate," Sci-Tech Information Development & Economy, vol. 5, 2007.

[10] LAN Xiaoji, QIU Juxiang, "Digital City GIS-Virtual Tourism," Land and Resources Informatization, vol. 1, 2009.

[11] LIN Yao-hua, ZHU Kun-zi, CHEN Jing, "The Development of Putian City Real Estate Information System Based on WEB GIS," Journal of Xinxiang University (Natural Science Edition), vol. 3, 2009.

[12] Le Hoang Son, Pham Huy Thong, Nguyen Duy Linh, Nguyen Dinh Hoa, and Truong Chi Cuong, "Some Results of 3D Terrain Splitting By 2D Polygonal Vector Data," International Journal of Machine Learning and Computing, vol. 1, no. 3, pp. 253-262, 2011.

[13] Le Hoang Son, Pham Huy Thong, Nguyen Duy Linh and Nguyen Dinh Hoa, "An Integration of Particle Swarm Optimization and Bread-Distributing Task for 3D Terrain Splitting Problem," International Journal of Computer Science & Emerging Technologies, vol. 2, no. 4, pp. 463 – 469, 2011.

[14] Le Hoang Son, Pham Huy Thong, Nguyen Duy Linh, Truong Chi Cuong and Nguyen Dinh Hoa, "Developing JSG Framework and Applications in COMGIS Project," International Journal of Computer Information Systems and Industrial Management Applications, vol. 3, pp. 108 – 118, 2011.

[15] Le Hoang Son, Pham Huy Thong, Truong Thi Hanh Phuc, Nguyen Dinh Hoa, Nguyen Thi Hong Minh, "Some Extensions of Terrain Splitting and Mapping Problem," International Journal of Computer Theory and Engineering, vol. 3, no. 5, pp. 590 – 597, 2011.

[16] Le Hoang Son, Pham Huy Thong, Truong Thi Hanh Phuc, Nguyen Dinh Hoa, Nguyen Thi Hong Minh, "An Improvement of SESA algorithm for Terrain Splitting and Mapping Problem," Journal on Information Technologies & Communications, Special Issue on Research, Development and Application on Information & Communication Technology (Vietnam), vol. 6, no. 26, pp. 271 – 279, 2011.

[17] Le Hoang Son, Pham Huy Thong, "A novel stochastic-based optimization algorithm for 3D terrain splitting by 2D polygonal vector data," Journal of Research and Practice in Information Technology, submitted for publication.

[18] M. van Kreveld, Digital Elevation Models: overview and selected TIN algorithms. Berlin: Springer-Velag, 1997.

[19] Miao Xuelan, "Application of vrml and webgis technology to the traveling geographic information system," Computer Applications and Software, vol. 7, 2005.

[20] MA Pengfei, ZHAO Wenji, HU Zhuowei, DUAN Fuzhou, CAI Wenbo, "Research of 3 - Dimensional Visualization of Rural Folk Tourism Sight," Geospatial Information, vol. 5, 2009.

[21] Nicholas Metropolis, "The beginning of the Monte Carlo method," Los Alamos Science, Special Issue, pp. 125–130, 1987.

[22] Nguyen Duc Thien, Le Hoang Son, Pier Luca Lanzi, and Pham Huy Thong, "Heuristic Optimization Algorithms For Terrain Splitting and Mapping Problem," International Journal of Engineering and Technology, vol. 3, no. 4, pp. 376 – 383, 2011.

[23] Ortiz, S., "Is 3D Finally Ready for the Web?," Computers, vol. 43, no. 1, pp. 14 – 16, 2010.

[24] SONG Wei, LI Hua, "Research of 3D Web GIS system based on X3D," Computer Engineering and Design, vol. 11, 2005.

[25] SHI Rong, XU Hui-ping, Chen Hua-gen, "Application of 3D Virtual WebGIS in Earthquake Disaster Prevention and Mitigation," Journal of Seismological Research, vol. 2, 2008.

[26] WANG Feng, LIU Ren-yi, LIU Nan, "Study on the application of WebGIS and virtual reality technology in tourism development," Journal of Zhejiang University (Sciences Edition), vol. 6, 2005.

[27] WANG Wei,WU Sheng, "Research on the Integration of 3D Geographic Information Web Services," Geomatics World, vol. 1, 2008.

[28] YIN Xue-song, "On Real Estate Management Informatization of PKU," Research and Exploration in Laboratory, vol. 3, 2009.

[29] ZHANG Yong-fu, LIU Jin-bao, MU Yang, "Analysis and Design of City Tourism Information System Based on WebGIS and Virtual Reality Technology," Aeronautical Computing Technique, vol. 1, 2008.

AUTHORS PROFILE

Le Hoang Son is a researcher at the Center for High Performance Computing, Hanoi University of Science, VNU. His major field includes Data Mining, Geographic Information Systems and Parallel Computing. He is a member of IACSIT and also an associate editor of the International Journal of Engineering and Technology (IJET). He also served as a reviewer for PACIS 2010, ICMET 2011, ICCTD 2011, International Journal of Computer and Electrical Engineering (IJCEE), Imaging Science Journal (IMS). Email: sonlh@vnu.edu.vn. Tel.: +84-904-171-284.