# Some Results of 3D Terrain Splitting by 2D Polygonal Vector Data

Le Hoang Son, Pham Huy Thong, Nguyen Duy Linh, Nguyen Dinh Hoa, and Truong Chi Cuong

***Abstract*—In this paper, we consider the problem of displaying large DEM terrains over the Web environment and present some new results of 3D terrains splitting for the benefit of displaying. Our methods are performed with the supports of 2D Polygonal Vector Data (2PVD) and contain two algorithms: 2OPS and SESA. The first one, 2-Objective Parallel Splitting Algorithm (2OPS), is proposed to split a given terrain into some small ones following by polygons in 2PVD. This algorithm is based on parallel computation and is designed for quick splitting process. Similarly, the second algorithm, Space Reduction Splitting Algorithm (SESA), is also used for terrain division but for a smaller memory space in each processor. Finally, evaluations of time and space complexity as well as a series of numerical experiments are performed to reveal some characteristics of our methods and prove their suitability for the original problem.**

***Index Terms*—2D Polygonal Vector Data, 2OPS, 3D WebGIS, Parallel Computation, SESA, Terrain Splitting.**

## I. Introduction

Geographic Information System on web environment or WebGIS is being used as an important tool for various applications in real life such as tourism[2], line management[4], simulation[5], agriculture[13], natural resource management[14], city information system[25], pipe network[26], and many other fields[11] [24] [27]. Its advantages can be seen as the capabilities to share and exploit information in map forms through networks. Then, important decisions may be made efficiently for the sake of saving costs and utilizing available material facilities.

The three dimensional WebGIS system is a higher development than previous GIS-2Ds and originated from the fact that people want to enhance the visualization of GIS. 3D WebGIS can provide realistic visualization of spatial information and has immense potential in infrastructure management (life-line and network infrastructure), civil construction, disaster management, 3D city simulation, and geological modeling, etc. Indeed, it is considered to be the main focus of GIS scientists[1].

These kinds of systems use Digital Elevation Model (DEM) and Digital Surface Model (DSM) data to represent terrains and living objects on them, respectively. Among them, DEM is the most important component because it shows the composition of a geographic area in a 3D form. It is generated by many methods, for example via satellites, air planes, LIDAR technology, etc. As stated in [16] [22] [28], DEM or Grid DEM consists of a matrix data structure with the topographic elevation of each pixel stored in a matrix node. Grid DEMs are distinct from other DEM representations such as Triangular Irregular Network (TIN) and contour based data storage structures. In general, the time to display Grid DEM is often faster than other kinds'. Thus, Grid DEMs are often used as sources in the process of 3D terrain generation and display.

However, sizes of Grid DEMs are often large depending on their resolution. For example, a 30m Grid DEM has a volume of 280 Megabytes (MB). The smaller the resolution of Grid DEM is, more details in 3D terrain are shown and its size is increased as a result. Thus, it takes long time to display terrains totally especially on web environment which requires fast processing in short time. A recent survey in [15] has shown that the maximal volume of DEM terrain for the fastest display on JSG which, in essence, is a 3D WebGIS system is approximately 1.2 MB. Comparing with the volume of 30m Grid DEM above, we can easily recognize that it is very difficult to display the terrain in such conditions. This problem should be overcome in order to make 'truly' 3D WebGIS systems in equivalent to what have been represented in 2D GISs and WebGISs.

To deal with this obstacle, our idea is to divide the original 3D terrain into some small ones for the benefit of displaying. Assume that we have a set of 2D Polygonal Vector Data (2PVD) in ESRI Shape formats[6] related to the DEM terrain. Then, the division is performed following by polygons in 2PVD to ensure the spatial characteristics between regions. This process is implicit performed by a script on the 3D WebGIS system after uploading the DEM terrain. Then, the display time of each small terrain is, of course, faster than the original one's. This solution may help 3D WebGIS systems reduce the possibility of being crashed or overload for processing large terrains. Additionally, it will focus users to each specific area for studying instead of the whole terrain.

In this paper, we will present some new results of the 3D terrains splitting by 2PVD problem. The first one, 2-Objective Parallel Splitting Algorithm (2OPS), is based on parallel computation techniques and is designed for quick splitting process. Similarly, the second algorithm, Space Reduction Splitting Algorithm (SESA), is also used for terrain division but for a smaller memory space in each processor. Finally, evaluations of time and space complexity as well as a series of numerical experiments are performed to reveal some characteristics of our methods and prove their suitability for the original problem.

Manuscript received May 30, 2011. This work is supported by a research grant of Vietnam National University, Hanoi for promoting Science and Technology.

Le Hoang Son is the corresponding author (e-mail: sonlh@ vnu.edu.vn).

The remainder of this paper is organized as follows. Section 2 elaborates some related researches. In Section 3 and 4, details of the algorithm 2OPS and SESA will be presented, respectively. The evaluations comprising of time and space complexity as well as numerical experiments are shown in Section 5. Finally, we make conclusion and future works in the last section.

## II. Related Works

Before we describe the main problem, let us begin with some definitions as following.

- ***Definition 1***: A Polygon $U_j$ is a sequence of 2D Points $M_i^j(x_i^j, y_i^j)$ where $i = 1,..,n_j$ and $n_j$ is total number of vertices in the polygon. It is oriented by a specific direction $Dt^j$. This polygon can contain holes defined as a ring. The sign (-) means the direction of polygon is counterclockwise. The direction of a ring is opposite to the one of polygon which consists of it.

- ***Definition 2***: A set of 2PVD can be expressed by a sequence $\{U_j / j = 1,..,l\}$ where $l$ is total number of polygons in 2PVD.

- ***Definition 3***: The area of a polygon in 2PVD is calculated as the area of the smallest rectangle containing that polygon. Similarly, the area of some polygons is the area of the smallest rectangle containing these polygons.

- ***Definition 4***: A rectangular grid in Digital Elevation Model is fixed and geographically specified by coordinate origins and the size in each cell $\langle (xc, yc), s \rangle$. Besides, the number of cells in this grid is defined as $m = nC \times nR$ where $nC$ is the number of columns and $nR$ is the number of rows.

- ***Definition 5***: The area of a 3D DEM terrain is defined as $nC \times nR \times s^2$. Additionally, the area a 3D DEM terrain in a processor is equal to the area of all polygons in that processor.

For our given problem, the traditional method is using a Rendering Engine[8] including Load, 3D Rendering and Transform steps. The first step which turns out to be the most time-consuming among all is dedicated for transferring 3D DEM terrains to clients' machines. Then, some 3D Rendering algorithms such as *Texture Mapping*[3], *Z-buffering*[3], *BSP tree*[7], *Photon mapping*[10], *Alpha Compositing* [19], *Pre-computed Radiance Transfer*[21], etc. are used to produce an image based on previously downloaded three dimensional data. The last step is used to attach geographic coordinate references to this image so that all points in the map will have coordinates. To serve for real time applications, for example WebGIS, an additional technique so called human eye perception is used. The idea is quite visual: showing as much information as possible as the eye can process in a fraction of a second, a.k.a. in one frame. As a result, the final image presented is not necessarily that of the real-world, but one close enough for the human eye to tolerate. All these techniques are currently applied in Geographic Virtual Markup Language (GeoVRML)[12] [23], KML[17], and GML[20] which are the most common geo-support standards to view 3D terrains on web. In other words, they enable geo-referenced data, such as maps and 3D terrain models, to be viewed over the web by users with a standard VRML plugin for their web browsers.

However, the limitation of these standards can be recognized as the requirement of downloading the whole DEM terrain before processing. Therefore, the larger the DEM terrain is, the longer the waiting time requests. Somehow, web browsers may fall into stuck or overload because of handling very large 3D terrains. Imagine that many users access a 3D WebGIS system and request the same terrain. Such these cases can cause full bandwidth in the connection line and, of course, increase the waiting time of users. For this reason, the above method is not suitable for processing large DEM terrains.

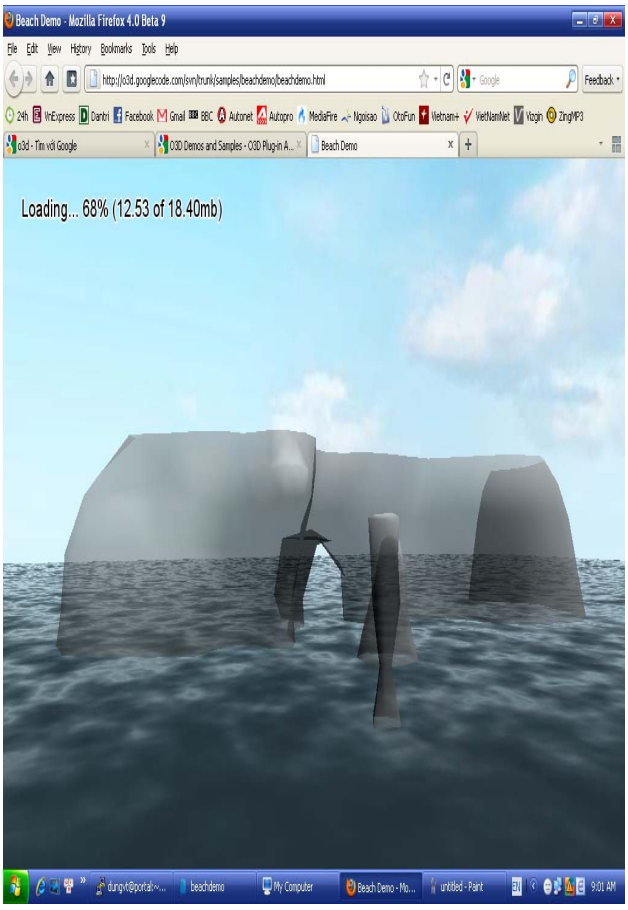

Fig. 1. 3D Scenes' Display in O3D

Another, recent approach has been presented by Google corporation namely O3D[18]. It is a JavaScript API for creating interactive 3D graphics applications that run in a browser window: games, 3D model viewers, product demos,

virtual worlds, etc. in real time. The idea behind this kind of standard is the capabilities of dividing the whole 3D scene into some parts and transferring them to clients one after another. Then, each part is constructed to display using the Painter's algorithm[8] or more advanced Z-buffering[3]. These algorithms paint distant parts of a scene before parts which are nearer thereby covering some areas of distant parts. While rendering a part, some other parts are transferred to client 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 (Fig. 1).

Nevertheless, O3D works with a special kind of 3D scenes. Instead of organizing it into a unique scene, a lot of small scenes are pre-created in the form of O3DTGZ. This means that the splitting process must not be performed with O3D. Besides, these small scenes are arranged in the order of distances. Indeed, an area in 3D scene can be divided into some small ones and spatial characteristic integrity is not ensured. Finally, O3D is used for 3D graphic applications and not designed for 3D GIS, especially DEM terrains. Anyway, the display time of O3D is still better than other methods and we can utilize parts of this idea for our objective.

Lastly, a recent study in [15] has shown a 'rough' method to split a 3D DEM terrain into some small ones by selected vertical and horizontal lines. Therefore, the display time is better than using traditional method. However, similar to O3D, spatial characteristics between regions are not kept. Thus, further analysis operations in a region such as area calculation, visibility, etc. can not be performed.

From all literatures above, we may see that 3D terrain splitting is the most suitable method for the original problem. However, instead of using arbitrary splitting methods, a set of 2PVD (Definition 1 and 2) is opted to support this process for the integrity of spatial characteristics between regions. As mentioned in the previous section, we use parallel computation techniques for this task. Hence, two major factors should be considered namely computing time and memory space in this approach. In the two next sections, details of these algorithms will be elaborated.

## III. The 2 - Objective Parallel Splitting Algorithm

The basic idea of this algorithm is to divide the original DEM terrain into some small ones following by the number of processors $k$ in the system and all polygons in 2PVD (Definition 1 and 2). This means that regions close together will be attached to the same processor.

Details of the *2-Objective Parallel Splitting Algorithm* (2OPS) are shown as below.

**Step 1**: For each polygon $U_j,\ j=1,..,l$ in 2PVD, find $x^j_{\max}, x^j_{\min}, y^j_{\max}, y^j_{\min}$

$$x^j_{\max}=\max\{x^j_i\} \text{ and } x^j_{\min}=\min\{x^j_i\} \quad , (1)$$

$$y^j_{\max}=\max\{y^j_i\} \text{ and } y^j_{\min}=\min\{y^j_i\} \quad , (2)$$

where $i=1,..,n_j$ and $n_j$ is total number of vertices in the polygon $U_j$.

**Step 2**: Find center values of the smallest rectangle containing the polygon $U_j,\ j=1,..,l$

$$C_j\left(\overline{x^j},\overline{y^j}\right)=\left(x^j_{\min}+\frac{x^j_{\max}-x^j_{\min}}{2}, y^j_{\min}+\frac{y^j_{\max}-y^j_{\min}}{2}\right) \quad . (3)$$

**Step 3**: Calculate the distance matrix

$$D=\left[d_{ij}\right]_{l\times l} \quad , (4)$$

where $d_{ij}=d(C_i,C_j)$ - Euclidean distance, $i=1,..,l$ and $j=1,..,l$.

**Step 4**: Each processor will have $\lfloor l/k \rfloor$ polygons. Additionally, the last processor should bear some extra polygons as the surplus of above quotient $l\%k$.

**Step 5**: For each processor, we traverse from the first unmarked polygon and mark its unmarked neighborhoods based on the distance matrix $D$ until $\lfloor l/k \rfloor$ polygons are reached. After this step, all polygons are arranged into processors.

**Step 6**: For each processor $h$, find the maximum and minimum of coordinates of all polygons belonging to it

$$X^h_{\max}=\max\{x^j_{\max}\} \text{ and } X^h_{\min}=\min\{x^j_{\min}\} \quad , (5)$$

$$Y^h_{\max}=\max\{y^j_{\max}\} \text{ and } Y^h_{\min}=\min\{y^j_{\min}\} \quad , (6)$$

where $h=1,..,k$ and $x^j_{\max}, x^j_{\min}, y^j_{\max}, y^j_{\min}$ are mentioned in Step 1.

**Step 7**: Convert four coordinates in Step 6 from the coordinate system of 2PVD to the ones in the coordinate system of 3D DEM terrain by means of the DEM's coordinate origins and size (Definition 4). Then, separate a rectangle in the original DEM terrain which contains all polygons in processor $h$ by these converted coordinates above. Then, save it as a small DEM terrain.

```
Program Separate (Output)
Begin
   Convert( X^h_min , X^h_max , Y^h_min , Y^h_max )
   for u = Y^h_min to Y^h_max do
   begin
     for v = X^h_min to X^h_max do
     begin
      if matrix[u][v] <> -9999 then
      begin
   splitComma(matrix[u][v],
& w1 , & w2 );
          if( w1 ) Output ( w1 ,  w2 );
          else Output ( w1 );
      else
         Output (matrix[u][v]);
      end;
     end;
   end;
End.
```

**Step 8**: Repeat Step 6 and 7 for other processors until all processors are processed.

The 2OPS method is a parallel terrain splitting algorithm with dedicated to computing time between processors. Because it assigns the same number of polygons to each processor, the number of tasks is definitely equal between processors except the last processor which has to bear the surplus $l\%k$. However, the number of polygons in 2PVD ($l$) is many times greater than the number of processor ($k$). Therefore, the extra job is inappreciable and we can assume that the dividing process assigns the same number of tasks to all processors. In theory, the parallel computing time of the algorithm is equal to the sequential computing time for original DEM terrain divided by the number of processors. Indeed, more processors we have, less waiting time is. Later, we will check this consideration through experiments in Section 5.

## IV. Space Reduction Splitting Algorithm

The 2OPS algorithm above can split the original 3D terrain into some small parts with the priority of computing time. However, a limitation of 2OPS which can be recognized at this time is the cost of memory space. In fact, 2OPS uses a lot of memories to store small terrains. Indeed, the worst case can happen when each small terrain's area is equal to original one's. Therefore, it takes $k \times O(m)$ in memory with $k$ is the number of processors and $m$ is the number of cells in original 3D terrain (Definition 4 and 5). This number can be hundreds or thousands Gigabytes. Consequently, the 2OPS algorithm is suitable for running in cluster servers where memory space is abundant, not in normal PC computers.

Our idea is to construct an algorithm to reduce the memory space for each processor. Moreover, this algorithm can be implemented to run in normal PC computers. First, let us specify the Input and Output of the problem. Assume that we have a 3D DEM terrain and some polygons in 2PVD as well as the number of processors $k$. We have to split the original terrain following by some polygons and the number of processors above and satisfying the conditions

**a) Condition $A_1$**: The area of the small DEM terrain in a processor is smaller than a given threshold $\alpha$ multiplying the area of original 3D terrain

$$\left|SP_i\right| \le \alpha \times S_{DEM} \text{, for } i = \overline{1,k} \quad . (7)$$

**b) Condition $A_2$**: The difference between two areas of terrains in two processors is smaller than a given threshold multiplying the area of original 3D terrain

$$\left|SP_i - SP_j\right| \le \varepsilon \times S_{DEM} \quad , (8)$$

for $i = \overline{1,k}$, $j = \overline{1,k}$ and $i \ne j$.

**c) Condition $A_3$**: Each polygon in 2PVD is fully contained in any processor.

To illustrate the requirements of our problem, let us consider two examples.

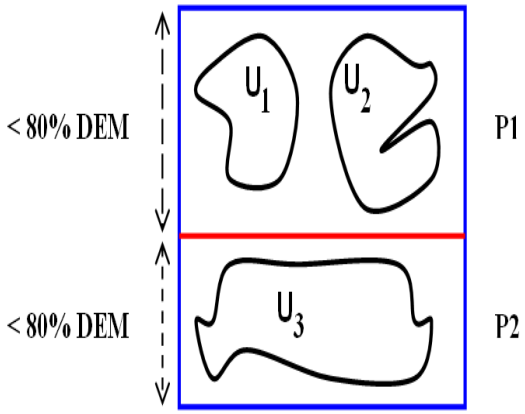


Fig. 2. A 2PVD with three polygons

Example 1:

Suppose that we have a 2PVD with three polygons (Fig. 2). The number of processors $k = 2$. A suitable way to split this 2PVD into two parts is drawing a horizontal line between polygons $U_1$, $U_2$ and $U_3$. Then, two small DEM terrains whose areas are $SP_1$ and $SP_2$ are created. With $\alpha = 80\%$ and $\varepsilon = 5\%$, we can easily check the conditions from $A_1$ to $A_3$.

- Condition $A_1$: $\left|SP_1\right| \le 80\% \times S_{DEM}$, $\left|SP_2\right| \le 80\% \times S_{DEM}$.

- Condition $A_2$: $|SP_1 - SP_2| \le 5\% \times S_{DEM}$.
- Condition $A_3$: It is easily recognized through Fig. 2.

Example 2: This case (Fig. 3) can not be divided in a normal way. Suppose that the number of processors $k = 2$. If we put polygons $U_1$, and $U_2$ into a processor and $U_3$ into another one then the area of 3D DEM terrain in a processor is equal to the one of original DEM terrain. Therefore, the total memory space in this case is $2 \times O(m)$.

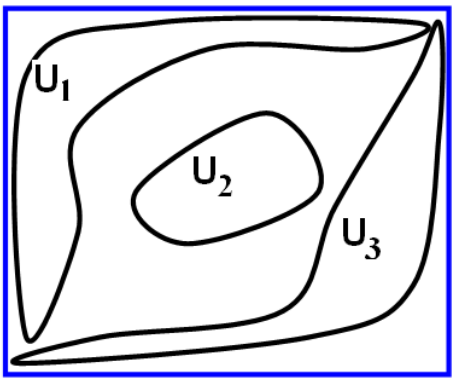


Fig. 3. An exceptional case

For a given, desired memory saving coefficient $\alpha$ of users, we have to find a suitable splitting way to reach that goal. Thus, an idea of conditional greedy partitioning algorithm is invoked. Basically, we traverse all partitions dividing $n$ elements into $k$ blocks. For each partition, calculate its area and check three conditions from $A_1$ to $A_3$. If finding one suitable partition, stop the algorithm and output the results. Certainly, to reduce the number of traversed partitions, a pre-processing step should be carried out to arrange some elements into specific blocks. For this reason, it is called a conditional greedy partitioning algorithm. The new algorithm is named as *Space rEduction Splitting Algorithm (SESA)*. Details of this algorithm are summarized as follows.

**Step 1**: Perform Step 1, 2 and 3 of 2OPS algorithm to calculate the distance matrix $D = [d_{ij}]_{l \times l}$ where $d_{ij} = d(C_i, C_j)$ is the distance between polygon $U_i$ and $U_j$ in 2PVD, $i = \overline{1,l}$, $j = \overline{1,l}$ and $i \ne j$.

**Step 2**: Based on the distance matrix $D$, find an unmarked polygons $U_i$ and its closet unmarked polygon $U_j$ in term of minimal distance and distance is smaller than $0.25 * \sqrt{nC^2 + nR^2}$ (Definition 4) until all polygons are reached.

**Step 3**: Calculate the area of polygon $U_i$ ($S_i$), $U_j$ ($S_j$) and both $U_i$ and $U_j$ ($S_{ij}$) (Definition 3).

Example 3:

$$S_j = \left(x^j_{\max} - x^j_{\min}\right)\left(y^j_{\max} - y^j_{\min}\right) \quad . (9)$$

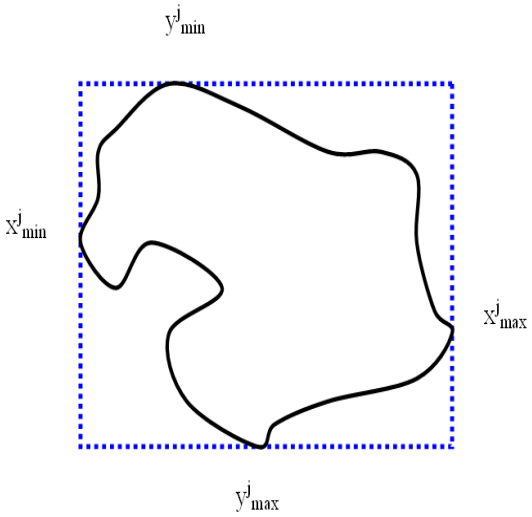


Fig. 4. Calculate area of a polygon

**Step 4**: (*Additive Condition*) If $S_i \ge 80\% S_{ij}$ or $S_j \ge 80\% S_{ij}$ then we add polygons $U_i$ and $U_j$ into a processor.

This is the pre-processing step before partitioning. Originated from two special cases below, we have designed a condition to reduce the number of traversed partitions.

- ***Case 1***: The smallest rectangle containing polygon $U_1$ consists of two smallest ones containing $U_2$ and $U_3$. Therefore, in this case, we should combine three polygons $U_1$, $U_2$ and $U_3$ into a single processor (Fig. 5).

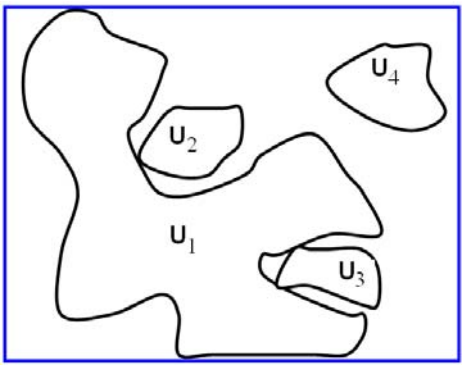


Fig. 5. Containment

- ***Case 2***: The area of polygon $U_1$ is greater than or equal to 80 percent of the area of two polygons $U_1$ and $U_2$. Therefore, we should also combine these two polygons into a single processor (Fig. 6).

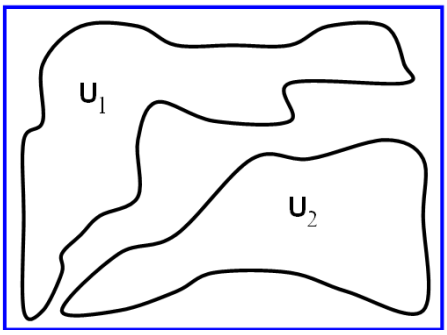

Fig. 6. Adjacency

**Step 5**: Repeat from Step 2 to Step 4 for other unmarked polygons. The final result is a set of polygons: $<\{U_i, U_j\}, U_h / i, j, h = \overline{1,l}; i \neq j \neq h>$ (*)

**Step 6**: Use a parallel partitioning algorithm to divide the set (*) into $k$ blocks with $k$ is the number of processors. In this case, we have used the best parallel partitioning algorithm from Hoang Chi Thanh et al. [9].

**Step 7**: For each received block $i$, calculate the area of all polygons in this block $SP_i$, $i = \overline{1,k}$.

**Step 8**: Check the conditions from $A_1$ to $A_3$. If they are satisfied then stop the partitioning algorithm and perform the Step 6, 7 and 8 of 2OPS algorithm for all current blocks. Otherwise, return to Step 6 to find another solution.

**Step 9**: In case of no partitions satisfying the original conditions, we conclude that for given parameters $\alpha$ and $\varepsilon$, there does not exist any solution for our problem. Therefore, if users want to find other solutions then they should adjust the parameters. For example, $\alpha' = \alpha + 5\%$ and $\varepsilon' = \varepsilon + 1\%$. In this situation, return to Step 6 to find other solutions.

Once a solution is found, the memory space in each processor is saved by $\alpha$ percents. Indeed, the total memory space over all processors is considerably reduced. A remark can be extracted from the Step 5 of this algorithm is that we are able to combine more polygons to form sets of three, four or higher number of polygons. However, it is not effective in terms of computational time. Because, we must re-compute the distance matrix and, perhaps, the number of clusters can be different from the number of processor $k$. To satisfy both space and time complexity, we stop at level two.

## V. Evaluations

In this part, we evaluate two algorithms above both by time and space complexity as well as numerical experiments.

In 2OPS algorithm, Step 1, 2 and 5 require $O(l)$ memory space and time complexity. Step 3 requires $O(l^2)$ memory space and time complexity. In Step 6, 7, and 8, the memory space in the worst case is $k \times O(m)$ and the time complexity is $O(\lfloor l/k \rfloor)$. Finally, the time complexity of 2OPS is $O(l^2)$ and the total memory space in the worst case is $k \times O(m)$.

In SESA, Steps 1 to 5 require $O(l^2)$ time complexity. The partitioning step requires $O(1)$ time complexity in the best case and in the worst case

$$\left[\frac{1-\alpha}{k \times \varepsilon}\right] \times S(l,k) \quad , (10)$$

Where $S(l,k)$ is the number of partitions that divide a set of $l$ elements into $k$ blocks. Therefore, the time complexity of SESA in the best case is $O(l^2)$ and in the worst case is $\left[\frac{1-\alpha}{k \times \varepsilon}\right] \times S(l,k)$.

Because the areas of small DEM terrains are equal to $\alpha \times O(m)$ and the memory space used to store the distance matrix is $O(l^2)$, the total memory space in SESA is $\alpha \times k \times O(m)$.

However, theoretical time complexity does not always indicate clearly the speed of an algorithm. For this, measurements of CPU time often give better information. Therefore, in this part, we have implemented the two proposed algorithms (2OPS and SESA) in C programming language and executed them on a Linux Cluster 1350 with eight computing nodes of 51.2GFlops. Each node contains two Intel Xeon dual core 3.2GHz, 2GB Ram.

First, we study the running time of 2OPS algorithm to split a fixed DEM terrain whose sizes are 4039 x 6529 with four processors following by different number of polygons (Fig. 7). The result shows that the running time changes slightly when the number of polygons is smaller than 1000. For instance, when the number of polygons increases tenfold, the average increment of the running time is 2.85. In case the number of polygons is greater than 1000, the running time increases linearly. The average increment of the running time, in the same condition with above, is about 27.9. Therefore, we may predict the running time of 2OPS algorithm for a larger number of polygons through this test.

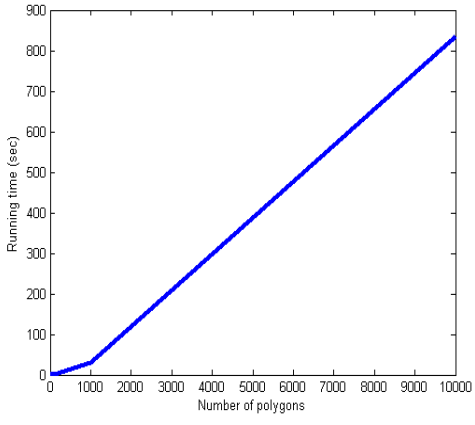

Fig. 7. The 2OPS's running time by number of polygons

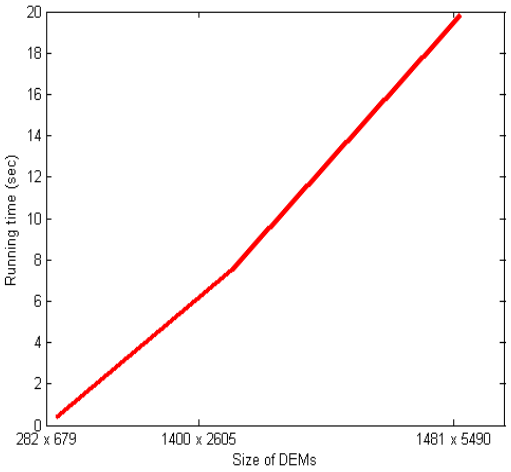


Fig. 8. The 2OPS's running time by size of DEMs

In Fig. 8, the running time of 2OPS following by various sizes of DEM terrains is shown. For example, when the size of a DEM terrain is 282 x 679 (about 191 thousands elevation values), the running time is 0.354 sec. However, contrary to previous test, the average increment of the running time is greatly changed where the number of elevation values is smaller than 4 million points or size of DEM terrains is lower than 2000 x 2000. In this situation, when the number of elevation values increases nineteen times, the running time is 21.4 times greater. This number, in remain case, is approximately 2.64 times greater. Indeed, the running time seems to be stably changed after exceeding the threshold above.

In the next test, we will investigate the running time of 2OPS when splitting a DEM terrain whose sizes are 4039 x 6529 following by the number of processors and the number of polygons (Fig. 9). It is obvious that the running times when using 2, 3 or 4 processors is similar and perhaps smaller than when using 1 processor for small cases which are under 200 polygons. Otherwise, more processors are provided, less computational time is required. In average, the running time is 1.6 times smaller per processor. Again, this test reconfirms that 2OPS is good in terms of computational time.

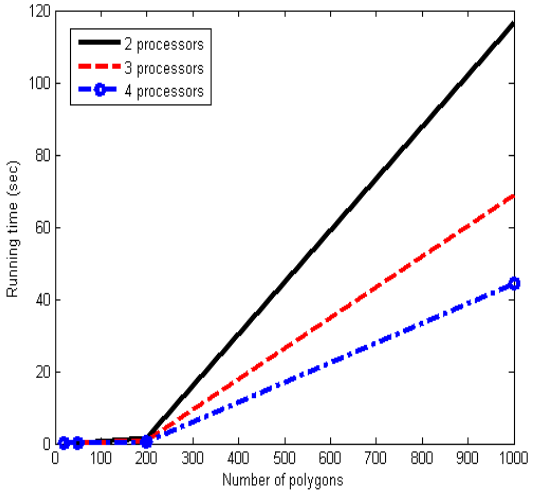


Fig. 9. The 2OPS's running time by number of processors and polygons

Fig. 10 shows the relation between the total (CPU) time spent on the performance of a program, the speedup and the efficiency of this performance. The speedup of the performance is defined as $S = T_s / T_p$, where $T_s$ ($T_p$) is serial execution time (parallel execution time), respectively. The efficiency of the performance is determined as $E = S / k$, where $k$ is the number of processors. Indeed, we can determine the number of optimal processors when running the 2OPS algorithm in a specific size of DEM and number of polygons by using the Efficiency and Speed up line, in this case, is three.

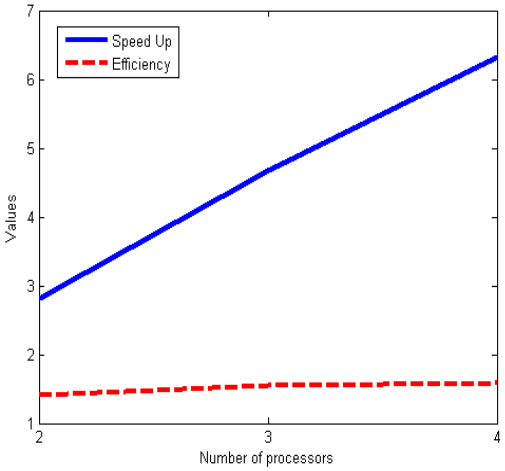


Fig. 10. Speed up and Efficiency of 2OPS

In Fig. 11, we use SESA algorithm for a given DEM terrain where the number of processors is four. This figure shows the relation between two parameters $\alpha$ and $\varepsilon$ following by the number of polygons. For example, with 20 polygons in 2PVD, if $\varepsilon = 0.61\%$ then the required memory space is $1\% \times O(m)$. It is the minimal error threshold which means no partition is found if the error threshold is smaller than this number. In general, this figure provides a relative reference of how to choose the parameter $\varepsilon$ in order to reach the memory saving percent $\alpha$ in a given terrain. A similar test is illustrated in Fig. 12 when the number of processors varies.

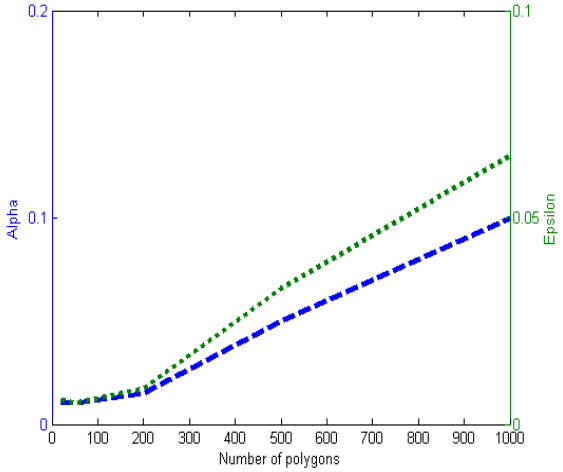


Fig. 11. Relation between $\alpha$ (Alpha) and $\varepsilon$ (Epsilon) in SESA algorithm

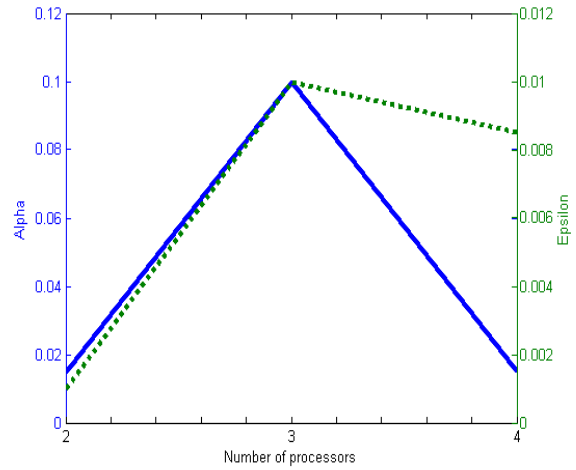


Fig. 12. Relation between $\alpha$ (Alpha) and $\varepsilon$ (Epsilon) by number of processors

In this part, we investigate the experimental parameter $\varepsilon$ of SESA with four processors and 200 polygons and $\alpha = 25\%$ following by various sizes of DEM terrains (Fig. 13). The maximal number $\varepsilon = 18.9\%$ seems to be our recommended error threshold when running the SESA algorithm. Certainly, more experiments are needed to obtain the correct parameter for users' dataset. However, for some cases, the number above is the most suitable.

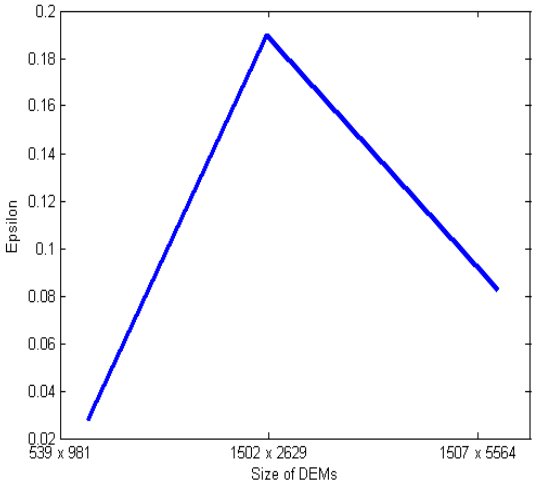


Fig. 13. The experimental $\varepsilon$ (Epsilon)

Finally, we compare the running time of 2OPS algorithm with the splitting method described in [15] (Fig. 14). The sizes of DEM terrain are 4039 x 6529. The parameter *Density* is defined as the average number of processor per polygons. The result shows that the 2OPS algorithm is faster than the splitting method, about 1.5 times in average. The reason may come from the way to split DEM terrains. In splitting method, terrains are divided without concerning spatial characteristics. Therefore, it has to keep all values in small DEM terrains. Conversely, 2OPS processes the smallest rectangles containing some polygons in equivalent to 2PVD only. Indeed, in most cases, the total area processed by 2OPS is less than the one divided by the splitting method. Thus, the running time of 2OPS is faster than the splitting method's. From this test, we may conclude that our method obtains fast processing in most cases while still keeping spatial characteristics between regions.

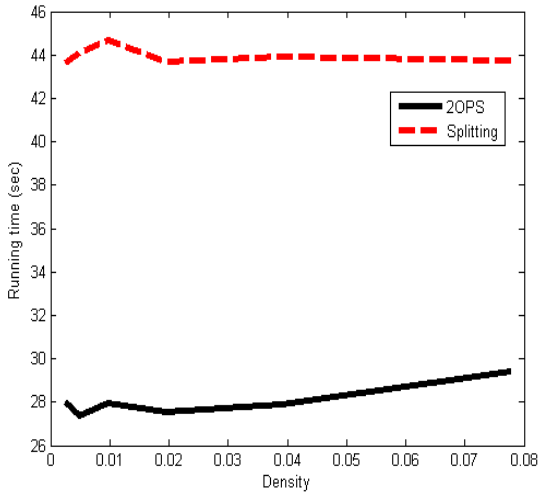


Fig. 14. Compare the running times of 2OPS and the splitting method [15]

## VI. Conclusion and Future Works

In this paper, we concentrate on the 3D Terrain Display problem in Geographic Information Systems. Throughout a brief introduction, we have shown how importance this problem can bring in practical applications as well as some difficulties that are currently faced when processing it over the Web environment. One of the most popular methods to deal with these obstacles is terrain splitting that is described in Section 2. Along with some state-of-the-art works in that section, our approach, based on parallel computing, can be considered as the amelioration of the best, current splitting method in [15] by keeping spatial characteristics between regions in 2PVD. Then, further advance analysis actions in each small terrain can be fully performed without difficulty. In details, two specific algorithms in our approach designed for computing time increment and memory space reduction in each processor namely 2OPS and SESA are presented. They are both evaluated by time and space complexity as well as numerical experiments. Some remarks in each test reveal characteristics of our methods and prove their suitability for the original problem. Lastly, we have implemented a simple 3D WebGIS in association with these algorithms to display 3D DEM terrains (Fig. 15).

In the future, we will investigate some heuristic algorithms for the SESA method. Besides, a complete 3D WebGIS system which combines advance GIS and 3D capabilities is also our mission. Eventually, we also study an effective method to store terrains in databases as well as fast displaying terrains through parallel computation.

## Appendices

The implementations and test datasets of these algorithms can be found at this address: http://chpc.vnu.vn/gis/tsm.rar

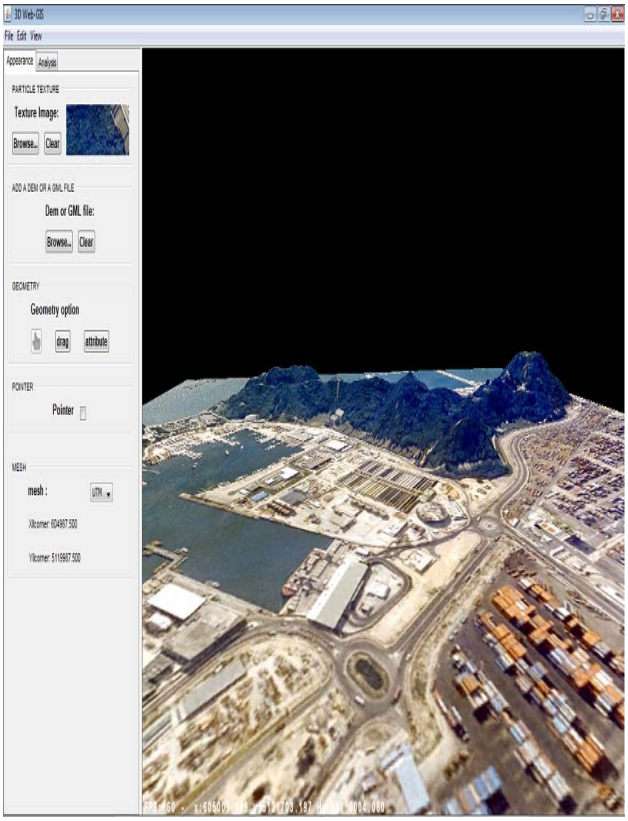


Fig. 15. A simple 3D WebGIS

## ACKNOWLEDGMENT

The authors are greatly indebted to Prof. Pham Ky Anh, VNU; Prof. Pier Luca Lanzi, Dr. Roberto Colonello, Politecnico di Milano, Italy and anonymous reviewers for their comments and suggestions which improved the quality and clarity of paper. Another thank will be sent to Prof. Hoang Chi Thanh, VNU who gave us a valuable paper for this research.

This work is supported by a research grant of Vietnam National University, Hanoi for promoting Science and Technology.

## REFERENCES

[1] A.A. Rahman, Zlatanova, S. and M. Pilouk, "Trends in 3D GIS development," *Journal of Geospatial Engineering*, vol. 4, no. 2, pp. 1-10, 2002.

[2] BAI Ming-zhe, DING An-min, ZHANG Jian-xiong, "Construction of Jiaozuo Tourism Information System based on WebGIS," *Journal of Jiaozuo Institute of Technology*, vol. 5, 2005.

[3] Catmull, E, "A subdivision algorithm for computer display of curved surfaces," PhD. dissertation, University of Utah, 1974.

[4] Cui Dong, Zhang Zhe, "A WebGIS Based Transmission Line Management System," *Power System Technology*, vol. 6, 2002.

[5] Chen Guohua, Zhang Jing, Zhang Hui, Yan Weiwen, "Chen Qingguang. Application of WebGIS to non-heavy gas cloud diffusion simulation," *Natural Gas Industry*, vol 26, no. 10, pp. 140-143, 2006.

[6] *Esri Shapefile Technical Description*, ESRI White Paper, 1998.

[7] Fuchs, H., Kedem, Z.M., Naylor, B.F, "On visible surface generation by a priori tree structures," In *Proceedings of SIGGRAPH*, vol. 14, no. 3, 1980, pp. 124–133.

[8] Foley J, Van D A, Feiner S K, Hughes J F, *Computer Graphics: Principles and Practice*, Reading, MA, USA, Addison-Wesley, 1990, pp. 1174.

[9] Hoang Chi Thanh and Nguyen Quang Thanh, "An efficient parallel algorithm for the set partition problem," *New Challenges for Intelligent Information and Database Systems*, Springer-Verlag, 2011.

[10] Jensen, H.W., Christensen, N.J, "Photon maps in bidirectional monte carlo ray tracing of complex objects," *Computers & Graphics*, vol. 19, no. 2, pp. 215–224, 1995.

[11] Lu Hongli, Wu Gang, "Development of a forest resource spacial information system based MAPINFO," *Journal of Beijing Forestry University*, vol. 3, 2001.

[12] LIU Jia-fu, LIANG Yu-hua, "Application Research of GeoVRML-technology on the 3D Geo-information Visualization," *Jilin Normal University Journal (Natural Science Edition)*, vol. 1, 2009.

[13] Le Hoang Son, "A WebGIS application in agricultural land management," *VNU Journal of Science, Natural Sciences and Technology*, vol. 25, no. 4, pp. 234 – 240, 2009.

[14] Le Hoang Son, Nguyen Quoc Huy, Nguyen Tho Thong and Tran Thi Kim Dung, "An effective solution for sustainable use and management of natural resources through WebGIS Open Source and Decision-Making Support Tools," In *Proceeding of the 5th International Conference on GeoInformatics for Spatial-Infrastructure Development in Earth and Allied Sciences (GIS-IDEAS) 2010*, Hanoi, Vietnam, December 9-11, 2010, pp. 87 – 92.

[15] 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 (IJCISIM)*, vol. 3, pp. 108-118, 2011.

[16] M. Albani, B. Klinkenberg, D. W. Andison, J. P. Kimmins, "The choice of window size in approximating topographic surfaces from Digital Elevation Models," *Int. J. Geographical Information Science*, vol. 18, no. 6, pp. 577–593, 2004.

[17] Mike Botts, George Percivall, Carl Reed and John Davidson, "OGC Sensor Web Enablement: Overview and High Level Architecture," *GeoSensor Networks*, vol. 4540, pp. 175-190, 2008.

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

[19] Porter, T., Duff, T, "Compositing digital images," *Computer Graphics*, vol. 18, no. 3, pp. 253–259, 1984.

[20] Ron Lake, "The application of geography markup language (GML) to the geological sciences," *Computers & Geosciences*, vol. 31, no. 9, pp. 1081-1094, 2005.

[21] Sloan, P., Kautz, J., Snyder, J, "Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low Frequency Lighting Environments," *Computer Graphics*, vol. 29, pp. 527–536, 2002.

[22] Tarboton, D.G, "A new method for the determination of flow directions and upslope areas in grid digital elevation models," *Water Resources Research*, vol. 33, pp. 309-319, 1997.

[23] THORNE, C (May 2011). Next Steps for X3D Geospatial Specification. Available: http://www.web3d.org/x3dearth/meetings/2007November/NextStepsForGeospatial.pdf

[24] WEN Huai-xing et al, "Research on the Mechanism of Agricultural Information Management System Based Web," *Journal of Anhui Agricultural Sciences*, vol. 6, 2005.

[25] Wang Guangqong, "The Design and Development of GIS-Based City Information System," *Natural Science Journal of Harbin Normal University*, vol. 1, 2006.

[26] Xiao Bai,Yu Hai Long, "A Designing and Implementing of Water-supply Pipe Network Information System Based on the MapXtreme Application Server," *Geography and Territorial Research*, vol. 3, 2002.

[27] XUE Li-xia, WANG Zuo-cheng, "Application of WebGIS in intelligent community management system," *Journal of Chongqing University of Posts and Telecommunications (Natural Science)*, vol. 5, 2006.

[28] Zhang, W. and D. R. Montgomery, "Digital Elevation Model Grid Size, Landscape Representation, and Hydrologic Simulations," *Water Resources Research*, vol. 30, no. 4, pp. 1019-1028, 1994.

**Le Hoang Son** is a researcher at the Center for High Performance Computing, Hanoi University of Science, VNU. He is a member of IACSIT and also member of the editorial board of the International Journal of Engineering and Technology (IJET). His major field includes Data Mining, Geographic Information Systems and Parallel Computing. Email: sonlh@vnu.edu.vn

**Pham Huy Thong** is a researcher and Master student at the Center for High Performance Computing, Hanoi University of Science, VNU. His research interests include Geographic Information Systems and Molecular Dynamics Simulation. Email: thongph@vnu.edu.vn

**Nguyen Duy Linh** is a researcher and Master student at the Center for High Performance Computing, Hanoi University of Science, VNU. His research interests include Geographic Information Systems and Grid Computing. Email: duylinh@vnu.edu.vn

**Nguyen Dinh Hoa** is an associate professor and vice director of the Information Technology Institute, VNU. His research areas include linear programming, optimization, data structure and algorithms, and Geographic Information Systems. He is member of the organizing committees of many prestigious national conferences since 1998. Email: hoand@vnu.edu.vn

**Truong Chi Cuong** is a collaborator at the Center for High Performance Computing, Hanoi University of Science, VNU. His research interests include Geographic Information Systems and Software Engineering. Email: truongchicuongbk54@gmail.com