Some Extensions of Terrain Splitting and Mapping Problem

Le Hoang Son, Pham Huy Thong, Truong Thi Hanh Phuc, Nguyen Dinh Hoa, and Nguyen Thi Hong Minh

***Abstract*—Terrain Splitting and Mapping (TSM) problem was originally proposed by the authors in [12]. It was designed to increase the terrain display and query capabilities from a large terrain dataset. This problem is very important in 3D WebGIS and has been being considered as one of some main focuses in the following year. Indeed, the aim of this note is presenting some extensions of the TSM problem. Then, two novel algorithms based on Simulated Annealing and Greedy algorithms are described. The Experiment section in this paper reveals some interesting results and promises to efficiently support to the original problem.**

***Index Terms*—3D WebGIS, greedy algorithms, simulated annealing, TSM problem.**

I. Introduction

One of the most challenging problem in 3D Geographical Information System (3D GIS) that are currently faced by many GIS scientists is the capabilities to query and display terrains. Depending on how large the resolutions of DEM terrains are [4], [7], [13], [14], [15], [16], [18], their volumes or sizes are increased as a result. Therefore, it is impossible to view these terrains in Web environment which require processing in short time. Assume that we are equipped a *2D Polygonal Vector Data* (2PVD) in association with the DEM terrain, our purpose is to map each polygon in 2PVD into equivalent area of the terrain. Besides, this DEM terrain should be divided into some small ones that contain one or more polygons in 2PVD.

The requirements above are exactly *Terrain Splitting and Mapping* (TSM) problem which was originally proposed by the authors in [12]. This problem is very important in creating 3D WebGIS systems and handling large DEM terrains. Moreover, as stated in [8], [9], [10], [11], it is considered to be the main focus of GIS scientists in the following years.

The problem is elaborated as follows.

Assume that we have a 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

- **Condition A_1**: The area of a small DEM terrain in a processor is smaller than a given saving threshold α multiplying the area of original 3D terrain: $|SP_i| \le \alpha \times S_{DEM}$, for $i = \overline{1,k}$.
- **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: $|SP_i - SP_j| \le \varepsilon \times S_{DEM}$, for $i = \overline{1,k}$, $j = \overline{1,k}$ and $i \ne j$.
- **Condition A_3**: Each polygon in a 2PVD is fully contained in any processor.

Our objective function is formulated as

$$J_1 = \sum_{i=1}^{k} SP_i \to \min \tag{1}$$

These are some parameters that we need to concern: two input parameters (ε, k) and an output result α. Initially, these parameters are assigned by some specific values. Then, the SESA algorithm [12] which, in essence, is a conditional greedy partitioning algorithm is used to sweep over all possible partitions and try to find a suitable one. Certainly, some geometric conditions are used to limit the iteration steps. If no solution is found with these parameters, new ones are generated and the above process is repeated until maximal allowable parameter changing steps is reached or a solution is found.

However, as we can see above, the parameters are initialized randomly and re-generated many times until an acceptable solution is found. In case of no solution in all steps, the answer time is quite slow. Obviously, this limitation is unacceptable and should be overcome. In fact, if we can choose the correct parameters then we can quickly conclude whether a solution may exist or not.

Consider two extensions of the TSM problem following by the number of processors k and the parameter ε. The first one looks for the optimal pair of parameters (α, k) with a given parameter ε. Similarly, the second one finds the optimal pair of parameters (α, ε) with a given number of processors k. Obviously, if we can find these optimal parameters then they can be used immediately to find a suitable partition in SESA algorithm without re-generation. In the other words, they are the correct parameter that we have mentioned above.

In this paper, we will present two novel algorithms for these extensions. One of them is based on Simulated Annealing and designed for optimization of (α, k). The rest is based on Greedy algorithm and used for optimization of

Manuscript received March 16, 2011; revised September 26, 2011. This work is supported by a research grant of Vietnam National University, Hanoi for promoting Science and Technology.

All authors are with the Vietnam National University, 334 Nguyen Trai, Thanh Xuan, Hanoi, Viet Nam

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

(α, ε) . Both these algorithms will be tested through experiments and checked the suitability for the original problem.

The remainder of this paper is organized as follows. Section 2 presents a method based on Simulated Annealing to find the optimal pair of parameters (α, k) . A greedy algorithm to find the optimal pair of parameters (α, ε) is presented in Section 3. Some experiments are carried out in Section 4. Finally, we will make conclusion and future works in the last section.

II. Extension 1: Choosing Optimal (α, k)

Although the number of processors is often determined beforehand following by the system's conditions and the available processors, somehow it is chosen by experience. Indeed, some operations are done ineffectively. We can not assume that more processors lead to less computational time. In fact, this consideration is not totally correct because more processors mean the number of groups increase. Thus, more time is spent to traverse new partitions. Of course, size of terrain in each processor is smaller in this situation. However, the cost of computational time is what we must pay for that benefit.

Our goal in this section is choosing a suitable number of processors that makes the output parameter α become smallest if possible. Formally, given a fixed parameter ε_0 , we have to find the pair (α, k) where $k \in [2, K_0]$ is number of processor and K_0 is the maximal, allowable number of processors. Normally, K_0 is set to $\left\lceil \frac{N}{2} \right\rceil$ where *N* is the number of polygons in 2PVD

$$J = c_1 \times \sum_{i=1}^{k} SP_i + c_2 \times k \to \min \tag{2}$$

With constraints

$$\begin{cases} |SP_i| \le \alpha \times S_{DEM} \\ |SP_i - SP_j| \le \varepsilon_0 \times S_{DEM} \\ 2 \le k \le K_0 \\ i = \overline{1,k}; j = \overline{1,k}, i \ne j \end{cases} \tag{3}$$

In the formula (2), c_1 and c_2 is adjusted factor

$$\begin{cases} c_1 \in [0,1]; c_2 \in [0,1] \\ c_1 + c_2 = 1 \end{cases} \tag{4}$$

We begin with the following theorem.

Theorem 1: The smallest value of parameter saving threshold α is equal to the quotient between areas of the maximal polygon and original terrain

$$\alpha \in [\alpha_t, 1] \text{ where } \alpha_t = \frac{SP_t}{S_{DEM}} \tag{5}$$

Proof:

Suppose that we have N polygons and each polygon belongs to one processor. In this case, the number of processor is equal to N . Additionally, if the area of polygon t (SP_t) is the largest among all polygons' area, then we obtain

$$\alpha_t = \frac{SP_t}{S_{DEM}} \ge \frac{SP_i}{S_{DEM}} = \alpha_i \tag{6}$$
$$\forall i = \overline{1,k} \text{ , } i \ne t$$

Therefore α_t is chosen to be the final saving threshold in this partition.

Assume that we have another partition. Because the number of processors is less than the number of polygons, some polygons will be definitely merged together in a same processor. These are two cases for this situation

- The areas of processors that contain many polygons are smaller than SP_t.

In this case, α_t is still the maximum among all α_i $\forall i = \overline{1,k}$, $i \ne t$. Therefore, it is chosen to be the final saving threshold in this partition.

- The areas of processors that contain many polygons are larger than SP_t.

Thus, there exists a processor having maximal area among all blocks (SP_0). Then, its parameter α_o is also chosen and

$$\alpha_o \ge \alpha_t \tag{7}$$

Eventually, the chosen parameter for this partition is greater or equal to α_t .

Similarly, we do the same process for other partitions and find a conclusion

$$\alpha \in [\alpha_t, 1]$$

Our idea for the problem (2) – (4) is using the Simulated Annealing to find the optimal pair of solutions. *Simulated annealing* (SA) [1], [6], [17] is a generic probabilistic meta-heuristic for the global optimization problem locating a good approximation to the global optimum of a given function in a large search space. Its idea comes from the annealing in metallurgy. Indeed, each step of the SA algorithm replaces the current solution by a random "nearby" solution, chosen with a probability that depends both on the difference between the corresponding function values and also on a global parameter temperature, that is gradually decreased during the process. The dependency is such that the current solution changes almost randomly when the temperature is large, but increasingly downhill as it goes to

zero. The allowance for uphill moves potentially saves the method from becoming stuck at local optima.

The algorithm to find the pair (α, k) based on simulated annealing is presented as follows.

SA-TSA (N, C_1, C_2, K_0, ε_0 , T_0, Δt , temp, m)

1. Calculate the parameter α_t following by (5).
2. Initialize a pair of random parameters (α, k) where $\alpha \in [\alpha_t, 1]$ and $k \in [2, K_0]$. Record this pair

$$(\alpha_1, k_1) \leftarrow (\alpha, k)$$
$$(\alpha', k') \leftarrow (\alpha, k)$$

3. Calculate the value of this pair

$$value1 \leftarrow value \leftarrow f(\alpha, k) = c_1 \times \alpha + c_2 \times k$$

4. While $T_0 > temp$

a) Find another neighbor pair of parameters (α', k') by random control

If (rand ( ) > 0.5)
If $(\alpha > \alpha_t)$
$\alpha' \leftarrow rand(\alpha_t, \alpha)$
Else
$\alpha' \leftarrow rand(\alpha_t, 1)$
End If
Else
$k' \leftarrow rand(2, K_0)$
End If

b) Create m random partitions to split a set of N elements into k' block by using *Stick Procedure* and random alternative permutation.

$Find\_Stick(n, k, i)$
If $k = 0$ *then stop*
$Stick(i) \leftarrow Rand(1, n-k+1)$
$Find\_Stick(n - Stick(i), k-1, i+1)$
End Procedure

$Create\_Partitions(m, N, k')$ (8)
$h \leftarrow 1$
$Find\_Stick(N, k', 1)$
Check Constraints
If (constraints are not satisfied)
While $h \le m$
$h \leftarrow h + 1$
Find two random blocks in the current partition.
If (Number of element in one of two blocks above contains more than an element)
Find a random element in this block
Move this element to remainder block
End If
Check Constraints
If (constraints are satisfied) then stop
End While
End If
Store the suitable partition if exist
End Procedure

c) If at least a random partition satisfies the constrains (3) then

i. Calculate

$$\Delta \leftarrow f(\alpha', k') - f(\alpha, k) \quad (9)$$

ii. If $\Delta < 0$ then

$$(\alpha, k) \leftarrow (\alpha', k') \quad (10)$$
$$f(\alpha, k) \leftarrow f(\alpha', k')$$

iii. Otherwise, if $e^{\frac{-\Delta}{T_0}} > rand()$ then re-calculate (α, k) and its value by the formulae (10)

iv. If $f(\alpha, k) < value$ then trace it

$$value \leftarrow f(\alpha, k)$$

Store this partition (*)

d) Reduce the temperature

$$T_0 \leftarrow T_0 \times \Delta t \quad (11)$$

5. If $value1 = value$

a) Create m random partitions to split a set of N elements into k_1 blocks similar to Step 4b.

b) If at least a produced partition satisfies the constraints (3) then stop and conclude that the pair (α_1, k_1) is outputted results with the value $f(\alpha_1, k_1)$ and a partition above.

6. Otherwise, conclude the pair (α, k) is outputted results with the value $f(\alpha, k)$ and a partition (*).

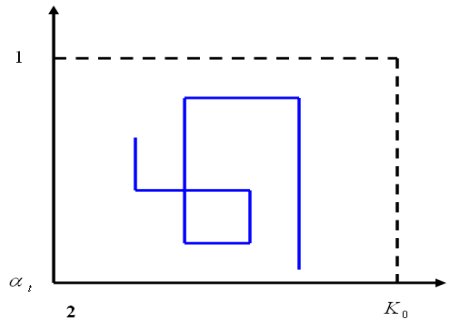

Fig. 1. A solution path

In case of no partition is satisfied after all, the program will return the default parameters $(\alpha, k) = (1, K_0)$. Fig 1 illustrates a solution path by example.

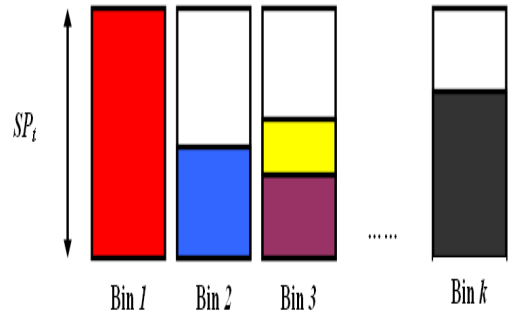

Fig. 2. The Bin Packing problem when α is fixed

A special case of the problem (2) – (4) can be found when we fix the parameter $\alpha = \alpha_t$. Indeed, if we assign the maximal polygon which holds α_t value to the first bin and other polygons to remainders bin with a constraint that area of these bins is smaller than or equal to SP_t then this problem turns back to the Bin Packing problem. Therefore, we can use *First Fit decreasing* [2], *Next Fit decreasing* [5] or *Best Fit* algorithm [3] for this problem. However, the pair (α, k) is not optimal.

Another strategy for this special case is using Binary Search for a suitable parameter k. However, it is far from optimum and similar to the Bin Packing's algorithm, should not be applied for the problem (2) – (4).

In practical, some parameters should be initialized as follows

- $C_1 = 0.6;\ C_2 = 0.4$
- $K_0 = 40$ processors
- $\varepsilon_0 = 2\%$
- $T_0 = 400$ (degree)
- $\Delta t = 0.999$
- $temp = 0.001$
- $m = 1000$ partitions

With these configurations, a large enough number of tests will help us find optimums efficiently.

III. Extension 2: Choosing Optimal (α, ε)

Given a number of processors k, this extension aims to find the smallest outputted parameter α with a 'loose' parameter ε. This means we do not pay much attention to the value of ε, but the smallest value of α instead. Indeed, we still examine the TSM problem in this extension with a slight change of the parameter ε above.

In contrast to the SESA algorithm [12], the new one should response quickly to a request. Hence, its complexity has to be linear instead of power. Therefore, a greedy algorithm is suitable for this requirement.

Similar to previous extension, we begin the algorithm with the following theorem.

Theorem 2: With all SP_i ($i = \overline{1,k}$) that satisfy three conditions from A_1 to A_3 then we obtain

$$\varepsilon \le \alpha \tag{12}$$

Proof:

First, we easily recognize that

$$\max\left\{\frac{SP_i}{S_{DEM}}; i = \overline{1,k}\right\} \ge \max\left\{\frac{SP_i - SP_j}{S_{DEM}}; \ i \ne j; i, j = \overline{1,k}\right\} \tag{13}$$

Follow the conditions A_1 and A_2 about α and ε, we obtain the formula (12).

Theorem 2 motivates us to use a 'loose' parameter ε as above. Because ε never exceeds α, indeed if the outputted parameter is smallest then ε is still in acceptable range. This range is found at [12] when the authors pointed out the relationship between specific parameters α and their equivalent range of parameters ε.

The idea of this greedy algorithm is to select k representative polygons having maximal areas and insert them into k blocks, one by one. Then, for the remainders, each polygon is put into a block that makes its total area be smallest among all possible coupling of other blocks (best fit). Therefore, the parameter α will be smallest if possible.

Summarization of the greedy algorithm is described as follows.

R-SESA (N, k)

1. Select k polygons, which have maximal area, from $Polygon[]$ which is a set of N polygons.
2. Insert these k maximal polygons into the partition P_i and mark these polygons.

$$P_i = \{Polygon[j_i]\};\ i = \overline{1,k}$$

3. Repeat the following steps from 3a to 3e until all polygons are marked.
   a. Select an unmarked polygon $Polygon[j]$
   b. Calculate each parameter α_s, $s = \overline{1,k}$ for each partition by the formula (13) when inserting that polygon into it.
   c. Find the partition which has minimal parameter α_s among alls. Assume that it is P_l.
   d. Insert $Polygon[j]$ into P_l

$$P_l = P_l + \{Polygon[j]\}.$$

   e. Mark $Polygon[j]$.

Example: With $N = 5$, $k = 2$ and the area of each polygon is

$$SP_1 = 100;\ SP_2 = 300;\ SP_3 = 150;$$
$$SP_4 = 50;\ SP_5 = 250;\ S_{DEM} = 1000$$

The following steps illustrate our algorithm.

Step 1: Two maximal polygons are $Polygon[2]$ and $Polygon[5]$.

Step 2: Make the first partition and mark $Polygon[2]$ and $Polygon[5]$.

$P_1 = \{Polygon[2]\}$ $P_2 = \{Polygon[5]\}$

Fig. 3.The first partition

Step 3: Select an unmarked polygon $Polygon[1]$ and find the suitable partition to put it into.

$$\alpha = \min\left\{\max\left\{\frac{SP_2 + SP_1}{S_{DEM}}; \frac{SP_5}{S_{DEM}}\right\}; \max\left\{\frac{SP_2}{S_{DEM}}; \frac{SP_5 + SP_1}{S_{DEM}}\right\}\right\}$$
$$\Rightarrow \alpha = \frac{SP_5 + SP_1}{S_{DEM}} = 0.035$$

Step 4: Make the second partition and mark the polygon $Polygon[1]$.

$P_1 = \{Polygon[2]\}$ $P_2 = \{Polygon[5], Polygon[1]\}$

Fig. 4. The second partition

Step 5: Repeat Step 3 and Step 4 for the unmarked polygon $Polygon[3]$ and receive the third partition

$P_1 = \{Polygon[2]\ Polygon[3]\}$ $P_2 = \{Polygon[5],\ Polygon[1]\}$

Fig. 5. The third partition

Step 6: The last partition is

$P_1 = \{Polygon[2]\ Polygon[3]\}$ $P_2 = \{Polygon[5],\ Polygon[1]\ Polygon[4]\}$

Fig. 6. The last partition

Step 7: All polygons are marked. The algorithm is stopped with $\alpha = 4\%$ and $\varepsilon = 0.2\%$.

A quick evaluation of time complexity of this algorithm can be seen as follows. The total running time is equal to the sum of time to find k representatives and to select a best block. The first task requires $O(N * k)$ and the second one consumes $O((N-k) * k)$ time complexity. Because the number of processors k is often much smaller than the number of elements N, indeed the total time complexity is approximate $O(N)$. This time complexity is equal to the one in SESA algorithm [12] in best cases. Therefore, this algorithm can be used for initial estimation before running SESA algorithm.

IV. Experiments

In this section, we have implemented the proposed algorithms SA-TSA and R-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. These algorithms were run against a large DEM terrain whose resolution is 25m and it contains more than 24 million elevation points. The 2PVD and DEM data are originated from Bolzano-Bolzen province, Italy in 2005 [11].

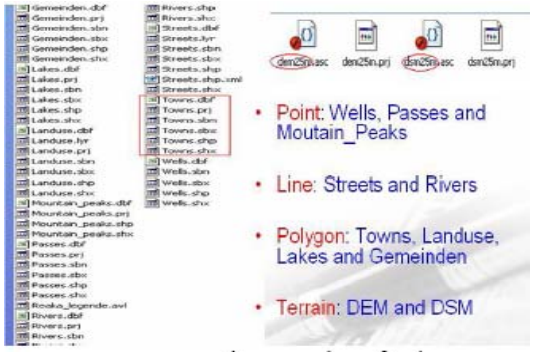

Fig. 7. GIS data of Bolzano-Bolzen province [11]

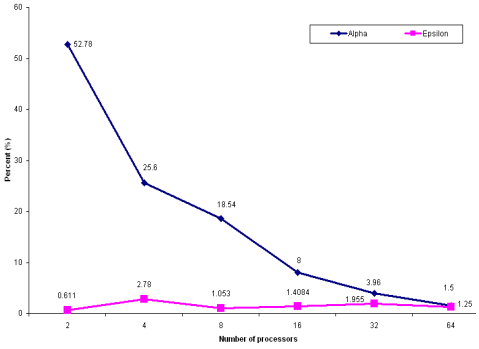

Fig. 8.Towns Polygon Shape

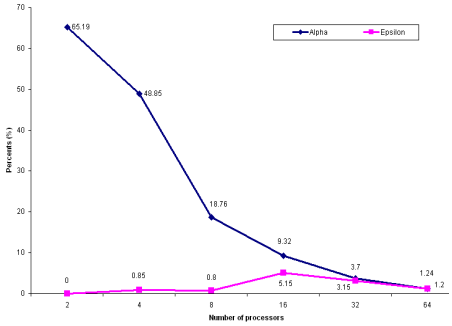

Fig. 9. Lakes Polygon Shape

First, we find the relationship between two parameters α (Alpha) and ε (Epsilon) when running the R-SESA algorithm with three polygon shapes in 2PVD: Towns, Lakes and Gemeinden following by the number of processors (Fig 8, Fig 9, and Fig 10). Obviously, for each terrain, the values of α are inversely proportional to the number of processors.

This means when the number of processors increases, the value of α reduces. Moreover, the parameter ε varies depending on types of terrain.

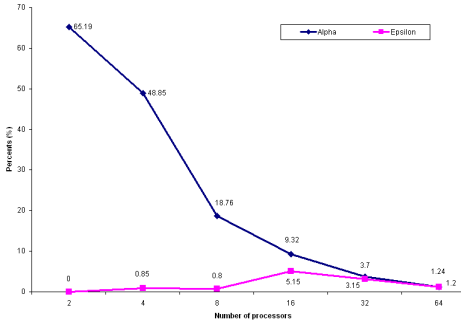

Fig. 10. Gemeinden Polygon Shape

For sparse terrains having many polygons, for instance Towns (821 polygons - Fig 8), maximal value ε is 2.87%. But with dense terrains having small number of polygons- Lakes (347 polygons - Fig 9) and Gemeinden (116 polygons- Fig 10), the maximal value ε increases to 5.97%. However, these values are still better than the ones in SESA algorithm when running with similar polygons and number of processors. To obtain the trade-off between these two parameters, we should set the number of processors about 64.

Second, we compare the running time, α and ε (Fig 11) of R-SESA and SESA algorithms [12] when the number of polygons increases and the number of processors is eight.

	Number of Polygon	100	200	400	600	800
Time (ms)	SESA	1.22536	3.71602	8.24581	13.68561	18.71062
	R-SESA	0.01	0.01	0.02	0.02	0.02
α (%)	SESA	36.1	66.1	77.1	83.1	83.1
	R-SESA	3.808063	19.35742	11.48606	12.80846	18.54324
ε (%)	SESA	36.1	66.1	77.1	83.1	83.1
	R-SESA	2.41814	13.75	2.50761	2.099103	1.05338

Fig. 11. Compare R-SESA vs. SESA by number of polygons

From Fig 11, we recognize that the running time of R-SESA is much faster than the SESA. More polygons are added, more difference between two lines is shown. Similarly, the parameters α and ε of R-SESA is better than SESA. For example, the parameter α of R-SESA is about 10.54% to 29.28% of SESA's parameter. When the number of polygons is greater than 800, the difference between two lines seems to be stable, about 30%. Therefore, we may say that the R-SESA can reduce more memory space in each processor than SESA does. Besides, the parameter ε of R-SESA is also better than SESA's. When the number of polygons is 200, the difference between two lines is shortest among all. However, when this number is larger, two lines are far away from each other. In stable state, the parameter ε of R-SESA is about 1.26% of the one in SESA algorithm.

	Number of Processors	2	4	8	16	32
Time (ms)	SESA	8.5369	17.43926	21.35948	27.40629	38.3523
	R-SESA	0.02	0.02	0.02	0.03	0.03
α (%)	SESA	83.1	83.1	83.1	83.1	83.1
	R-SESA	52.7882	25.59172	18.543244	7.99854	3.958282
ε (%)	SESA	29.1	83.1	83.1	83.1	83.1
	R-SESA	0.6112	2.783923	1.05338	1.408413	1.955714

Fig. 12. Compare R-SESA vs. SESA by number of processors

Again, we compare the running time, α and ε (Fig 12) of R-SESA and SESA algorithms with Towns polygon shape when the number of processors increases.

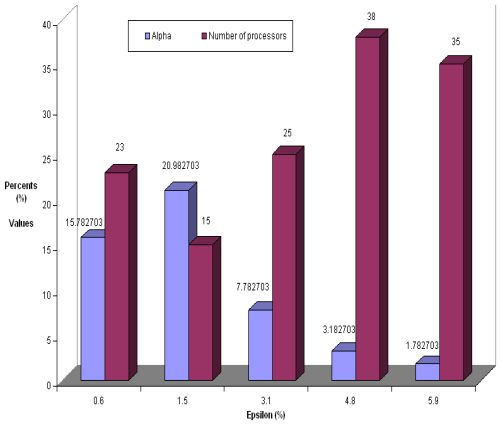

Fig. 13. Relationship of (α, k) in Towns Polygon Shape

Obviously, more processors lead to more partitions. Indeed, the running time of SESA algorithm increases as a result. Meanwhile, R-SESA seems not change too much. It always keeps the running time about several milliseconds. Furthermore, the α line of R-SESA tends to go down and reduce its value while the line of SESA seems unchanged. This shows the saving memory percent in R-SESA is better than SESA's while the parameter ε of R-SESA is many times smaller than SESA's.

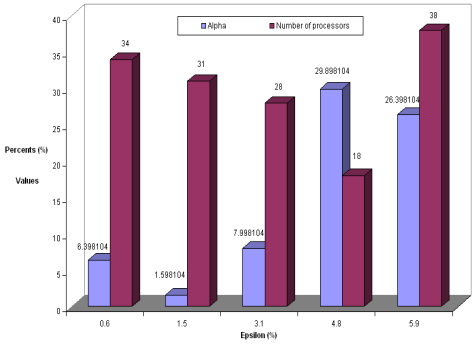

Fig. 14.Relationship of (α, k) in Lakes Polygon Shape

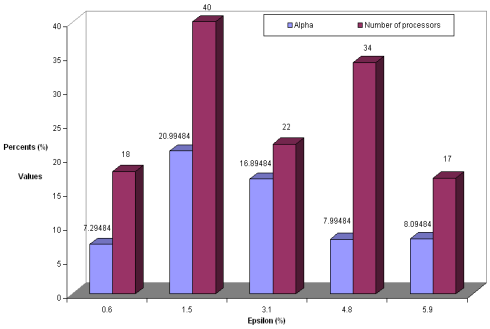

Fig. 15.Relationship of (α, k) in Gemeinden Polygon Shape

Fourth, we look for the pair of parameters (α, k) when running the SA-TSA algorithm with three polygon shapes in 2PVD: Towns, Lakes and Gemeinden following by the parameter ε - Epsilon (Fig 13, Fig 14, and Fig 15). The algorithm is run in the same configurations described in Section 2.

From Fig 13, we can easily recognize that when ε is smaller than 1.5%, the number of processors tends to go

down. Meanwhile, the parameter α increases. However, these two bars are totally reversed when $\varepsilon > 1.5\%$. This can be explained by the requisition of less partition to more processor for the adaptation of parameter ε. Moreover, the parameter α is getting smaller when ε increases.

Similarly, in Fig 14, the variation of these two bars is the same with the ones in Fig 13. However, only a minor change can be seen that the changing point increases to 4.8% instead of 1.5% as above.

In Fig 13 and Fig 14, we check the algorithm with sparse polygons (Towns and Lakes). However, the results seem to be different with dense polygons (Gemeinden- Fig 15). Indeed, these two bars are directly proportional to each other. However, when $\varepsilon > 4.8\%$, the parameter α does not reduce as usual. Instead, it increases by one percent when moving to the next step. Therefore, the parameter $\varepsilon = 4.8\%$ is the perfect threshold in this case.

Throughout three figures above, the best value of SA-TSA $\alpha = 1.59\%$ is recorded when $\varepsilon = 1.5\%$ of Lakes polygons (Fig 14). Comparing with R-SESA when the best result of α is from 1.24% to 4.58% at 64 processors, we may choose ε as the prior parameter than the number of processors.

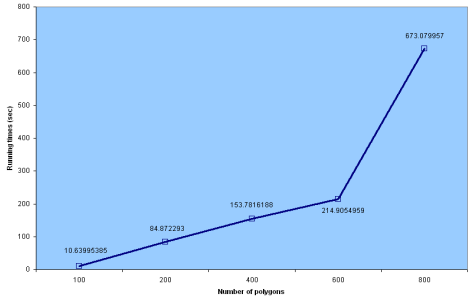

Fig. 16.Running times of SA-TSA by number of polygons

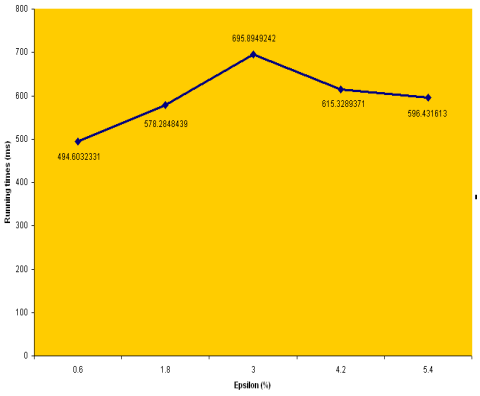

Fig. 17.Running times of SA-TSA by ε

Finally, we study the running time of SA-TSA algorithm following by the number of polygons (Fig 16) and parameter ε (Fig 17). From these results, we recognize that when the number of polygons increases, the running time is longer as a result. However, the increasing ratio is not equal for different number of polygons. For example, when this number changes from 100 to 200, the running time is 8 times longer. But if changing from 200 to 400, there is only 1.8 times. The decrement of increasing ratio shows the fact that when a solution is found in a specific number of polygons, the running time in that point is approximate to other later ones. Therefore, we can easily predict the running time of SA-TSA algorithm when $N > 800$.

Fig 17 shows other correlation of ε and the running time. This dependency is not direct proportional to the increment of ε. Normally, the range of this parameter is $(0,5)$. Below the value of three, the higher the parameter ε is, more time is required to process. The average increment of processing time in this range is approximately 18 percents. For the range $(3,5.4)$, the time tends to go down with average reduced ratio is about 10 percents. In fact, the longest time is located at $\varepsilon = 3$. Thus, for the benefits of time-saving and good saving threshold, we should choose the middle point of each range as the suitable value of ε.

The experimental results above have shown some properties of parameters of our algorithms.

V. Conclusion

In this paper, we have investigated some extensions of the TSM problem following by the number of processors k and the parameter ε. Throughout a brief introduction, we have shown how importance these extensions are for the original TSM problem. Then, two novel algorithms dealing with them are presented. The first one is based on Simulated Annealing and designed for the optimization of the pair (α, k). The remainder based on Greedy algorithms is used to find the pair (α, ε) in $O(N)$ time complexity. Both these algorithms are carefully tested through a lot of experiments and successfully proved the suitability for the TSM problem. Besides, they can be the references for further studies.

In the future, we will study an effective synthesis of these extensions for the original problem. Moreover, some further researches on the terrain parameters are also our task.

Acknowledgment

The authors wish to thank anonymous reviewers and the research group at the Center for High Performance Computing, VNU for supporting and giving useful comments. This work is supported by a research grant of Vietnam National University, Hanoi for promoting Science and Technology.

Appendix

The source code and test dataset can be found at this address: http://chpc.vnu.vn/gis/e-tsm.rar

References

[1] Amir Masoud Rahmani and Mojtaba Rezvani, "A Novel Genetic Algorithm for Static Task Scheduling in Distributed Systems," *International Journal of Computer Theory and Engineering* (IJCTE), vol. 1, no. 1, 2009, pp. 1-6.

[2] Brenda S. Baker, "A new proof for the first-fit decreasing bin-packing algorithm," *Journal of Algorithms*, vol. 6, issue 1, March 1985, pp. 49-70.

[3] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham, "Worst-Case Performance Bounds for Simple One-Dimensional Packing Algorithms," *SIAM J. Computer*, vol. 3, issue 4, 1974, pp. 299-325.

[4] 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.

[5] J. Csirik et al, "A probabilistic analysis of the next fit decreasing bin packing heuristic," *Operations Research Letters*, vol. 5, issue 5, November 1986, pp. 233-236.
[6] Kirkpatrick, S.; C. D. Gelatt, M. P. Vecchi, "Optimization by Simulated Annealing," *Science, New Series*, vol. 220, issue 4598, 1983, pp. 671–680.
[7] LIU Yongqiong, WU Yanlan, HU Hai, HU Peng, "Assessment method of digital elevation models accuracy and its limitations," *Journal of Geomatics*, vol. 5, 2009.
[8] Le Hoang Son, "On the Development of Three Dimensional WebGIS Systems: Some New Trends and Prospects," In *Proceedings of the 2010 3rd IEEE International Conference on Computer Science and Information Technology* (IEEE ICCSIT 2010), July 9 - 11, 2010, Chengdu, China, vol. 1, pp. 182 - 186.
[9] Le Hoang Son, "An Approach to Construct SGIS-3D: a Three Dimensional WebGIS System Based on DEM, GeoVRML and Spatial Analysis operations," In *Proceedings of the 2nd IADIS International Conference Web Virtual Reality and Three-Dimensional Worlds 2010* (IADIS Web3DW 2010), July 27 - 29, 2010, Freiburg, Germany, pp. 317 – 326.
[10] Le Hoang Son, "An Exploratory Study about Spatial Analysis Techniques in Three Dimensional Maps for SGIS-3D systems," In *Proceedings of the 2010 IEEE International Conference on Electronics and Information Engineering* (IEEE ICEIE 2010), August 1 - 3, 2010, Kyoto, Japan, vol. 1, pp. 199 - 203.
[11] 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, 2011, pp. 108-118.
[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* (IJMLC), vol. 1, no. 4, October 2011 (to be published).
[13] M. van Kreveld, "Digital Elevation Models: overview and selected TIN algorithms," In *Algorithmic Foundation of GIS*, M. van Kreveld, J Nievergelt, T. Roos and P. Widmayer, Ed., Springer-Velag, 1997.
[14] MA Chi, SONG Wei-dong, "Creating urban DEM by use of topographic map with DWG format," *Journal of Anshan University of Science and Technology*, vol. 5, 2004.
[15] Sandra Lanig, Arne Schilling, Beate Stollberg and Alexander Zipf, "Towards Standards-Based Processing of Digital Elevation Models for Grid Computing through Web Processing Service (WPS)," In *Proceeding of Computational Science and Its Applications* (ICCSA 2008), Lecture Notes in Computer Science, 2008, Volume 5073/2008, pp. 191-203.
[16] S. Rayburga, M. Thomsa and M. Neave, "A comparison of digital elevation models generated from different data sources," *Geomorphology*, vol. 106, issues 3-4, 15 May 2009, pp. 261-270.
[17] V. Cerny, "A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm," *Journal of Optimization Theory and Applications*, vol. 45, 1985, pp. 41-51.
[18] WANG Ke-ke, ZHANG Li-chao, PAN Zhen, WANG Qing-shan, ZHANG Shi-quan, "Research on Multiresolution Dynamic Creating Networks Algorithm of DEM based on DirectX," *Beijing Surveying and Mapping*, vol. 2, 2008.

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

Truong Thi Hanh Phuc is a Master student of College of Technology, VNU and an employee of FPT Software JSC. Her interest research is Geographic Information Systems. Email: phuctth@gmail.com

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

Nguyen Thi Hong Minh is a doctor and dean of training bureau at School of Graduate Studies, VNU. Her major researches include Parallel algorithms and Molecular Dynamics Simulation. So far, she has performed many important, major projects of VNU. Email: minhnth@vnu.edu.vn