Evolutionary Computation Approach for Spatial Workload Balancing

Ahmed Abubahia[1], Mohamed Bader-El-Den[2], and Ella Haig[2]

1 School of Psychology and Computer Science,
University of Central Lancashire,
Preston, United Kingdom,
AAbubahia@uclan.ac.uk,
2 School of Computing, University of Portsmouth,
Portsmouth, United Kingdom

Abstract. The growing demand for Geographic Information Systems (GIS) calls for high computation reliability to handle vast and complex spatial data processing tasks. A better parallel computing scheme should ensure balanced workload at different data processors to ensure optimal use of computing resources and minimise execution times, which poses more challenges with spatial data due to the nature of having spatial correlations and uneven distributions. In this paper, we propose a spatial clustering approach for workload balance, by using an evolutionary computation method that considers the nature of spatial data, to increase the computation performance for processing GIS polygon-based maps with massive number of vertices and complex shapes. To evaluate our proposed approach, We proposed two different experimental approaches for comparing our results: (i) Non–merging based experiment, and (ii) merging based experiment. The results demonstrated the advantage of the proposed spatial clustering approach in real GIS map based partitioning scenarios. The advantages and limitations of the proposed approach are discussed and further research directions are highlighted toward a development work by the research community.

Keywords: computational optimisation, geographic information system, spatial data, workload balancing, evolutionary computation

1 Introduction

As geospatial information become immense in demand, more reliable approaches are needed for achieving better processing performance. Integrating parallel computing and spatial analysis tasks provides a promising solution to the complexity of GIS data processing [13], [15], [24], [31], [32], [35], [36], [38].

Geospatial data comprises two model types [1]: raster data model and vector data model. The raster data represent geographical entities/features by a grid of

[1] http://www.esri.com/content/dam/esrisites/en-us/media/pdf/teach-with-gis/raster-faster.pdf

intensity pixels. The best example of raster data is satellite images. In contrast, the vector data represent geographical entities/features by a set of vertices and paths. There are three different shapes to represent the vector data, they are:

- Point – useful for representing features at small scale (e.g. bus stops)
- Line/Polyline – useful for representing linear or curved features (e.g. roads or rivers)
- Polygon/Area – useful for representing features at large scale (e.g. map of country)

This paper focuses on the vector polygon type of GIS data. The best parallel computing scheme should ensure balanced workload at different data processors to ensure optimal use of computing resources (i.e. distribute the workload equally to all processors, rather than having some overloaded processors and some idle ones), which poses more challenges with GIS data [8], [16], [30], [36], [39].

In the context of GIS vector data, the most known implementation examples are GIS-Hadoop [2] and Spatial-Hadoop [10]. They have been introduced as potential solutions for parallel spatial data processing. Although these systems have shown a good performance in terms of spatial data storage and query processing, they still lack the partition strategy that meets the workload balancing requirement. A particular challenge in most spatial analysis tasks is that a map polygon should be processed as a united structure that consists of a set of vertices. Each vertex can be considered as a tuple in database terminology. The workload balancing, in case of GIS data, could be met by partitioning the GIS map into groups of polygons where these groups should be approximately equal in terms of the total number of vertices, which is an optimisation challenge. A common heuristic approach for optimisation is the use of evolutionary computation algorithms, of which the most popular is the Genetic Algorithm (GA) approach. In this paper, we argue that the spatial workload balancing challenge could be solved by applying evolutionary computation to partition large GIS maps into groups of polygons. These groups should be approximately equal in terms of the total number of vertices, and we propose an evolutionary computation based approach that consider both the nature of spatial data and the workload balancing requirement to extend the computation reliability for processing GIS complex data.

The rest of this paper is organised as follows: Section 2 reviews previous work on GIS vector map partitioning. Section 3 introduces the proposed evolutionary based approach for balanced workload based GIS vector map partitioning. Section 4 describes the experiments, including the data used and the experimental setup for the evaluation of the proposed partitioning approach. Section 5 discusses the experimental results, while Section 6 concludes the paper and outlines directions for future work.

2 Background and Related Work

Geospatial information systems (GIS) are computer-based systems that facilitate the input, storage, manipulation and output of geographic location-based

data [27]. GIS data models are classified into two categories: raster and vector data models. outlines the different properties of vector and raster data. In GIS context, satellite images are the most known example of the raster model. GIS vector data, which is the focus on this paper, has three components: spatial data, attribute data and index data. Spatial data describes the map itself and always takes the form of three basic geometrical entities, which are: points, lines/polylines and polygons. Points are used to define a single location of an object; they are used to represent real-world objects, such as bus stops, traffic lights and street lights. Lines/Polylines define linear objects; they can range from two-point lines to complex strings that have many vertices; lines are used to represent real-world objects, such as rivers and roads. Polygons define area-based objects; they can range from rectangles to multi-sided shapes with many vertices; polygons are used to represent real-world objects, such as lakes, shopping areas, buildings and city boundaries.

All these map entities are formed by many organised vertices; spatial data is actually a sequence of coordinates of these vertices based on a certain geographical coordinate system. The most used format of GIS spatial data is the ESRI (Environmental Systems Research Institute) shape file. The ESRI shape file [12] has become an industry standard in geospatial data due to its compatibility, to some extent, with recently released GIS software products. The attribute data describes the properties of map entities through links to the location data. Attributes can be, for example, names or matching addresses. The most known example of GIS attribute data format is the ESRI database file that is associated with the ESRI shape file and needs to have the same prefix as the shape file [12]. Last but not least, in the GIS context, the index data describes a file structure, such as total file length, for either spatial or attribute data. The ESRI index file [12] is the best known example of index files. Large regional partitioning is the process of dividing a large geographic area consisting of spatial objects, i.e points, lines or polygons [22]. This paper focuses on the polygon type of map entities. Partitioning a large map into sub-sets of spatial entities is not an easy task due to the nature of having spatial correlations and uneven distribution. This problem has been investigated mostly in the redistricting field of GIS applications [4], [22], [28]. Some work has been done in the research of graph and GIS map clustering [5], [6], [7], [9], [11], [14], [20], [23], [26], [33], [34] and more attention given to the clustering of polygon-based type of GIS maps.

The previous approaches focus on attribute data rather than spatial data, and used evolutionary computation techniques for optimising the polygons' partitioning based on the attribute data, such as polygon area or polygon population [17], [18], [19], [21], [37].

According to the MapReduce model, the workload balancing can be only achieved by distributing equal chunks of data records (i.e. number of vertices) to the MapReduce processors [3], [8], [10], [25], [29].

Here the constraint is that the set of vertices that belong to the same polygon should not be separated in the mapping task (i.e. the first task of the MapReduce process). In contrast to the previous work, this paper focuses on spatial

properties of GIS vector data, and considers both the nature of spatial data and the workload balancing requirement to extend the computation reliability for processing GIS complex data. We propose an approach of workload balancing by using evolutionary computation in partitioning large GIS maps into groups of polygons which are approximately equal in terms of the total number of vertices (i.e. number of data records). We proposed two different experimental approaches for comparing our results: (i) Non–merging based experiment, and (ii) merging based experiment.

3 GIS Map Partitioning

This section outlines the main steps for implementing the proposed partitioning approach, including the map index computation, and applying the genetic algorithm to the problem of workload-balanced partitions in the context of spatial data.

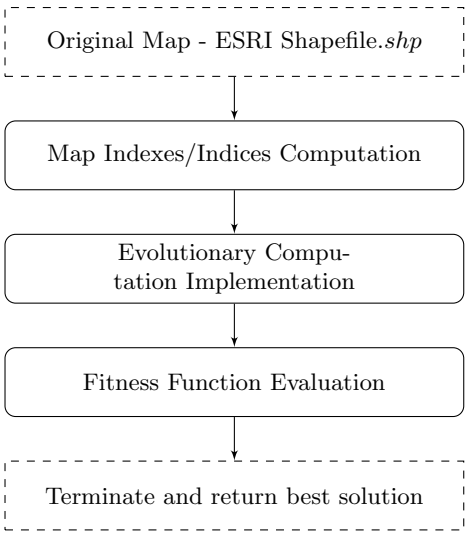


Fig. 1: The proposed evolutionary-based approach.

3.1 Map Indexes/Indices Computation

In the proposed approach, we argue that the use of polygons' representatives (indexes) will lead to faster processing rather than the use of the whole set of polygons' vertices. For identifying the map spatial features (polygons) indexes, we used the polygon's property of bounding box. Each polygon in the GIS vector map has a defined bounding box, which identifies the boundaries of each

polygon in the map; the coordinates for the bounding box are available in the shapefile [12]. The polygons' bounding box centres are calculated in both axes, as shown in Equation (1) and Equation (2), respectively.

$$x_c = \frac{x_{min} + x_{max}}{2} \tag{1}$$

$$y_c = \frac{y_{min} + y_{max}}{2} \tag{2}$$

where: x_c and y_c are the coordinates of polygon's centre in both x and y axes respectively; x_{min} is the minimum vertex coordinate in x-axis; x_{max} is the maximum vertex coordinate in x-axis; y_{min} is the minimum vertex coordinate in y-axis; y_{max} is the maximum vertex coordinate in y-axis. x_{min}, x_{max}, y_{min} and y_{max} are each of 8-byte length [12].

Algorithm 1: Genetic Algorithm

Data: polygon based map; Seed Population(POP_Size); crossover rate; mutation rate
Result: near-optimal balanced map partitions

START;
Initiate POP (POP_{size}) ;
Evaluate POP ;
while $GEN \leq GEN_{size}$ **do**
  for $i \leftarrow 1$ **to** $POP_{size} \times crossover_{ratio}$ **do**
    $Parent_1$ = Tournament Selection (POP, T_{size} ;
    $Parent_2$ = Tournament Selection (POP, T_{size} ;
    ($Child_1$, $Child_1$) = crossover($Parent_1$, $Parent_2$) ;
  end
  $POP_{new} \leftarrow Child_1$;
  $POP_{new} \leftarrow Child_2$;
  for $i \leftarrow 1$ **to** $POP_{size} \times mutation_{ratio}$ **do**
    $Parent_1$ = Tournament Selection (POP, T_{size} ;
    $Child_1$ = mutate($Parent_1$) ;
  end
  $POP_{new} \leftarrow Child_1$;
  $POP \leftarrow POP_{new}$;
  Evaluate POP ;
  $GEN++$;
end
Print best evolution/solution ;
STOP;

3.2 Evolutionary Computation Implementation

The Genetic Algorithm (GA) optimization technique is based on random search and has many advantages, such as performing search in complex and large spaces,

and providing near-optimal solutions. Unlike other optimization methods, GA is more suitable to this context of discrete variables based optimization problems.As shown in Algorithm 1, GA is a heuristic search algorithm that is based on evolutionary computation, which uses a random search to solve optimization problems. GA involves five main phases: initial population, fitness computation, selection, crossover and mutation. In Algorithm 1, POP refers to the population of individuals; POP_{size} represents the number of populations; GEN_{size} represents the number of generations; T_{size} is the number of tournaments. The initial population is randomly generated as a set of individuals, called a population, that represent solutions to the map partition problem. The parent selection is the process of selecting the fittest two pairs of individuals (i.e candidate solutions), based on standard deviation value, to be used in producing the next generation. As shown in Fig. 2, the crossover operator is the process of mating the selected parents to produce the next generation by identifying a crossover point. One crossover point is selected, and the coordinate values after the crossover point are copied from the second parent to the first child, and from the first parent to the second child. The mutation operator is responsible for maintaining diversity within the population and preventing premature convergence. As shown in Fig. 3, the selected coordinate value is inverted to a new coordinate value. The parent selection, crossover and mutation processes are carried for both horizontal lines (X–axis) and vertical lines (Y–axis), as shown in Fig 4.

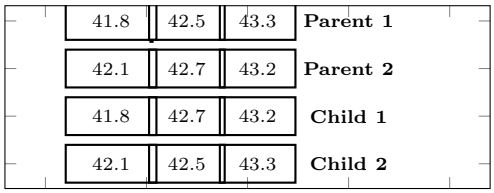


Fig. 2: Crossover Diagram Example

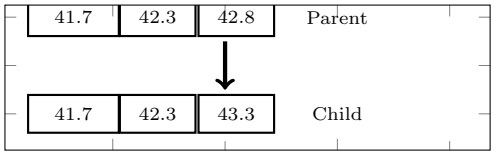


Fig. 3: Mutation Diagram Example

In the proposed approach, the resulted lines are combined to form the partitioning solutions, i.e. a set of three horizontal lines and three vertical lines would

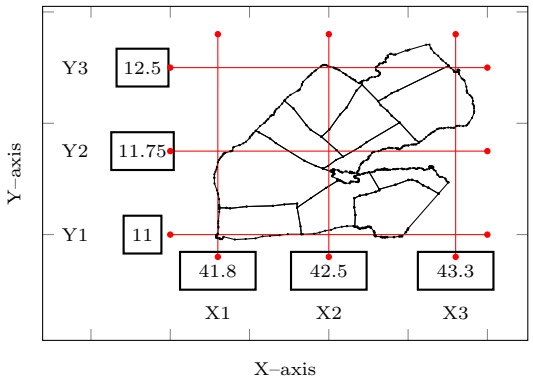


Fig. 4: Chromosome example of the horizontal and vertical solution lines (3×3 lines or 4×4 cells)

lead to sixteen (4×4) partitions. A collection of such solutions is called a population. The tournament selection method is used for selecting the parents, which has the advantage of diversifying the individuals set (i.e. candidate solutions). The process of parent selection, crossover and mutation continues iteratively for a specified number of generations until the fitness function is satisfied.

The fitness function for our problem is the standard deviation, as illustrated in Equation (3). The best solution is defined by the minimum standard deviation value.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N}} \tag{3}$$

where: σ is the standard deviation; x_i represents each value in the population; μ is the mean value of the population; and N is the number of values in the population.

The standard deviation is calculated at the level of the set of map partitions, where each set contain a number of polygons. The smaller the value of the standard deviation, the better the balance between the partitions according to the total number of vertices.

4 Data and Experiments

This section discusses the experimental setup for evaluating the performance and effectiveness of the proposed approach. Section 4.1 describes the data used in two sets of experiments; the initial set of experiments showed that some partitions had no vertices due to the uneven distribution of the data; consequently, an additional step was added to the partitioning approach to deal with these issues and a second set of experiments was carried out. Section 4.2 describes the initial experiments, while Section 4.3 describes the second set of experiments.

4.1 Data and Materials

We implemented our proposed approaches on a PC machine with the following specification: Windows–10 home premium 64-bits operating system, CPU 2.5GHz and RAM 8GB. The programming tasks has been implemented with Java version 8 update 171 in Netbeans integrated development environment.

To allow comparisons for maps of different sizes in terms of number of polygons and number of vertices, four datasets (of two maps each) combining high and low numbers of polygons and vertices were used, respectively:

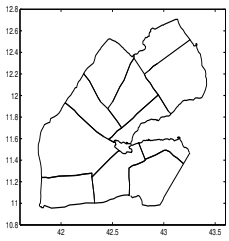
- **Dataset 1** includes maps with small number of polygons and small number of vertices.
- **Dataset 2** includes maps with small number of polygons and large number of vertices.
- **Dataset 3** includes maps with large number of polygons and small number of vertices.
- **Dataset 4** includes maps with large number of polygons and large number of vertices.

Within each dataset, the two maps are chosen to represent opposite ratios of number of polygons to number of vertices, i.e. one map has on average a smaller number of vertices per polygon compared with the other map in the same dataset. As shown in Table 1, eight GIS vector maps were used to implement our proposed approach, which are illustrated in Fig. 5, Fig. 6, Fig. 7 and Fig. 8.

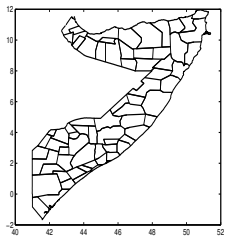
Table 1: The datasets with corresponding number of polygons, vertices and proportions of map size.

Dataset	Map	No. of Polygons	No. of Vertices	Average no. of vertices/polygon
1	Djibouti	11	676	61
	Somalia	88	3175	36
2	Guinea	56	21304	380
	Zimbabwe	81	32382	399
3	Liberia	305	10521	34
	Chad	347	19542	56
4	Burkina Faso	351	113996	324
	Ethiopia	575	261880	455

The used GIS maps are polygon-based maps that represent administrative boundaries of 8 countries in Africa, they are: Djibouti map of 11–polygons and 676–vertices (Fig.5a), Somalia map of 88–polygons and 3175–vertices (Fig.5b), Guinea map of 56–polygons and, 21304–vertices (Fig.6a), Zimbabwe map of 81–polygons and 32382–vertices (Fig.6b), Liberia map of 305–polygons and 10521–vertices (Fig.7a), Chad map of 347–polygons and 19542–vertices (Fig.7b), Burkina Faso map of 351–polygons and 113996–vertices (Fig.8a) and Ethiopia map of 575–polygons and 261880–vertices (Fig.8b). These vector maps are freely avail-

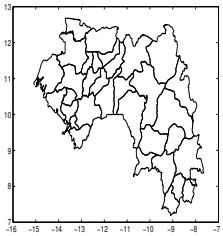


(a) Djibouti (11 polygons, 676 vertices)

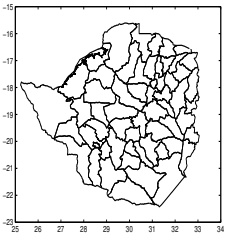


(b) Somalia (88 polygons, 3175 vertices)

Fig. 5: Data set 1.

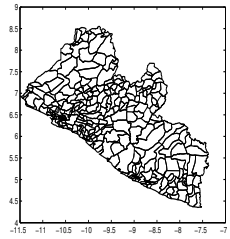


(a) Guinea (56 polygons, 21304 vertices)

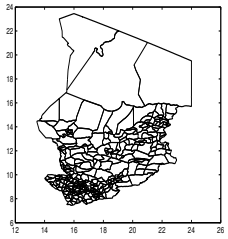


(b) Zimbabwe (81 polygons, 32382 vertices)

Fig. 6: Data set 2.

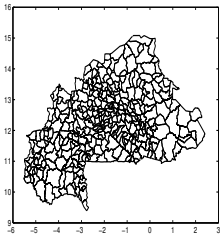


(a) Liberia (305 polygons, 10521 vertices)

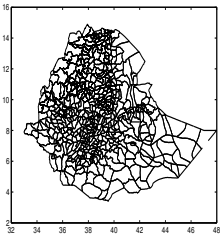


(b) Chad (347 polygons, 19542 vertices)

Fig. 7: Data set 3.



(a) Burkina Faso (351 polygons, 113996 vertices)



(b) Ethiopia (575 polygons, 261880 vertices)

Fig. 8: Data set 4.

able, in ESRI shapefile format[2], from the Map Library website[3]. ESRI Shapefiles (.shp) are considered as a popular format for geographic information system applications [1]. They have several key features: supporting point, polyline and polygon geometry formats, fast shape editing, easy reading and writing, small storage space, and storing both spatial/attribute information [12].

We ran an initial experiment which is described in Section 4.2 in which we found that some partitions had no vertices, especially for maps containing concave shapes. To address this issue, we introduce a merging step at each iteration. Consequently, we refer to the fist experiment as "non-merging" (described in Section 4.2) and the second experiment as "merging" (described in Section 4.3).

4.2 Non–merging based experiment

The experimental setup were as follows: both population size and generation number parameters were set to 10, 15 or 20, respectively. The crossover parameter was set to 0.8, the mutation parameter was set to 0.1, and the ratio parameter was set to 0.1. Moreover, each of these experiments was ran for 51 times – an odd number was chosen to have a median value as a data point rather than an average of the middle two data points. The standard deviation is used as the fitness function.

In our experiments, the number of grid cells (i.e. the number of partitions) were selected according to the number of polygons. For more clarification, the number of cells was set to 4×4 for maps with small number of polygons (i.e maps of Djibouti, Somalia, Guinea and Zimbabwe), while the number of cells was set to 6×6 for maps with a large number of polygons (i.e maps of Liberia, Chad, Burkina Faso and Ethiopia). The number of partitions can be user-defined to match the available number of processors in systems like MapReduce.

[2] http://www.esri.com
[3] http://www.maplibrary.org/library/stacks/Africa/index.htm

4.3 Merging based experiment

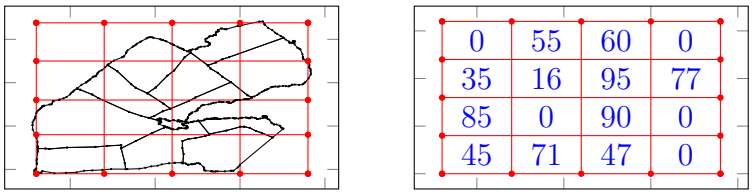
This approach follows the same implementation steps as in the non-merging experiment that were given above The only difference is that before computing the fitness function, a merging procedure is applied to the partitions based on a threshold value. In this paper, the average number of vertices per polygon is used as threshold value. This will be advantageous in avoiding the partitions that contain no vertices, i.e the number of vertices is equal to zero.

The threshold value (i.e. the number of vertices per cell) were selected according to the average number of vertices per polygon, and then set as follows: Djibouti Map (61 vertices), Somalia Map (36 vertices), Guinea Map (380 vertices), Zimbabwe Map (399 vertices), Liberia Map (34 vertices), Chad Map (56 vertices), Burkina Faso Map (324 vertices), and Ethiopia Map (455 vertices).

5 Results and Discussion

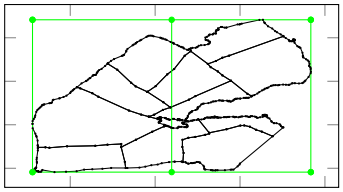
In this section the experimental results of the two sets of results are presented and discussed.

Fig.9 shows a solution example for the Djibouti map for both the non-merging and merging-based partitioning results for the experimental settings of: 51 run times, population size parameter of 20, generation number parameter of 20, crossover parameter of 0.8, mutation parameter of 0.1, and the ratio parameter of 0.1.

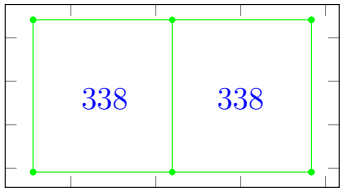


(a) Resulted Partitions (Non–merging Approach)

(b) No. of vertices per cell in (a)



(c) Resulted Partitions (Merging Approach)



(d) No. of vertices per cell in (c)

Fig. 9: Djibouti Map, examples of the best solution (chromosome)

In the non–merging experiment, as shown in the Figure 9 (a), there are 16–cells that represent the resulted partitions. Two partitions, i.e. upper-left and

upper-right corner cells, contain no vertices. Figure 9 (b) illustrates clearly how the number of vertices are distributed over the resulted partition based number of grid cells. In the merging experiment, as shown in the Figure 9 (c), there are 2 cells that represent the resulted partitions which should contain equal or nearly-equal number of vertices. In the shown figure each cell contains 338 vertices. Figure 9 (d) illustrate clearly how the number of vertices are distributed over the resulted partition based number of grid cells. To compare the results of the two sets of experiments, as well as the influence of the different values for the population size and the number of generations, we present the results in the form of box plots illustrating the range of values for the fitness function, i.e. the standard deviation. A box plot is a graphical shape for displaying the statistical range of values. Beginning from the top, the upper whisker represents the highest value in the range. Seventy-five percent (75%) of the resulted values fall below the upper quartile. The median marks the middle-point of the resulted standard deviation values and is shown by the line that divides the box into two parts. Half of the resulted values are greater than or equal to this value and half of the results have values lower than the median. Twenty-five percent (25%) of the resulted standard deviation values fall below the lower quartile. Finally, the lower whisker represents the smallest value in the range.

Figures (10 and 11) display the results for Dataset 1 against the population sizes of 10 and 20, respectively. Similarly, Figures (12 and 13) display the results for Dataset 2. Figures (14 and 15) illustrate the results for Dataset 3. Figures (16 and 17) show the results for Dataset 4.

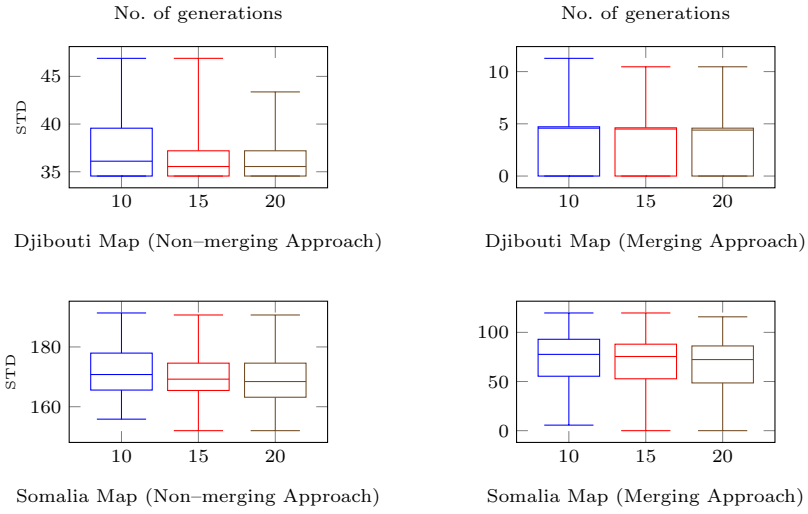


Fig. 10: Dataset 1, comparison between non-merging approach (left column) and merging approach (right column), No. of runs=51, pop size= 10

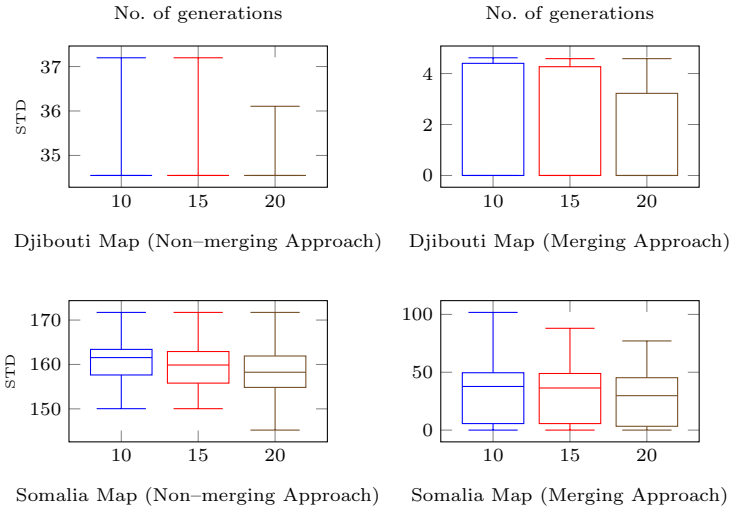


Fig. 11: Dataset 1, comparison between non-merging approach (left column) and merging approach (right column), No. of runs=51, pop size= 20

The experimental results show that an increase in the population size leads to lower values for the fitness criteria (standard deviation), which indicates better solutions, i.e. a more even distribution of the vertices among the partitions. Experiments with the Djibouti map, for example, show that when using the population size of 10, the standard deviation (STD) values range between 34.5 and 46.8 for the non–merging approach (top left in Figure 10), and between 0 and 11.2 for the merging approach (top right in Figure 10). For the population size of 20, the standard deviation values range between 34.5 and 37.1 for the non–merging approach (top left in Figure 11), and between 0 and 4.5 for the merging approach (top left in Figure 11).

For all population sizes (10 or 20 populations), the results show that the higher the generations number (10, 15 or 20 generations) for the reproduction process, the better the solutions produced, i.e. lower values for the standard deviation. This applies to all datasets regardless of the number of polygons or the number of vertices. Non–merging based experiments with the Liberia map (dataset 3), for example, show that in non–merging based experiments with population size of 20, when reproducing for 10 generations for the non–merging approach, the standard deviation values range between 203 and 318.6; for 15 generations, the standard deviation values are reduced to the range between 203 and 307.6; for 20 generations, the range is further reduced between 203 and 292.3; (top left in Figure 15. While when experimenting with the same map, show that in merging based experiments with population size of 20, when reproducing for 10 generations for the non–merging approach, the standard deviation values range between 116.6 and 284.4; for 15 generations, the standard deviation values are reduced to the range between 116.6 and 257.7; for 20 generations, the

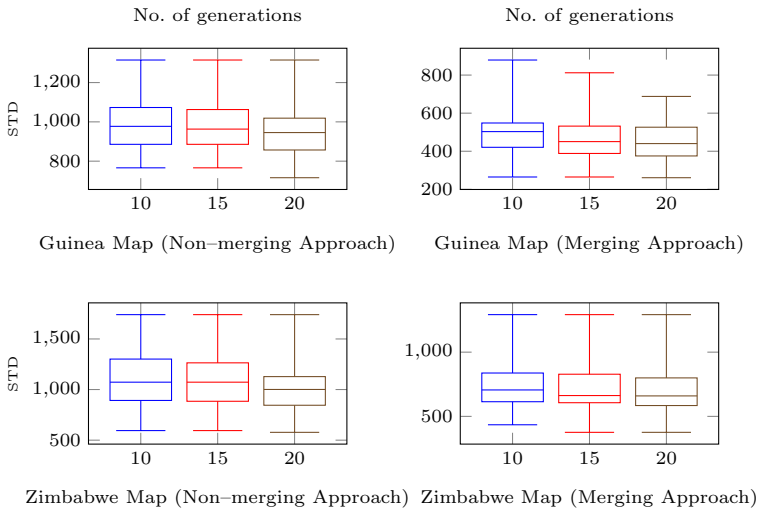


Fig. 12: Dataset 2, comparison between non-merging approach (left column) and merging approach (right column), No. of runs=51, pop size= 10

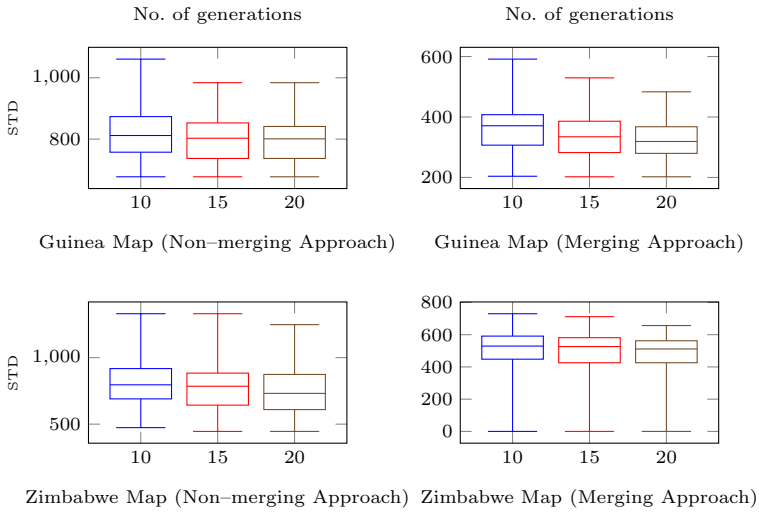


Fig. 13: Dataset 2, comparison between non-merging approach (left column) and merging approach (right column), No. of runs=51, pop size= 20

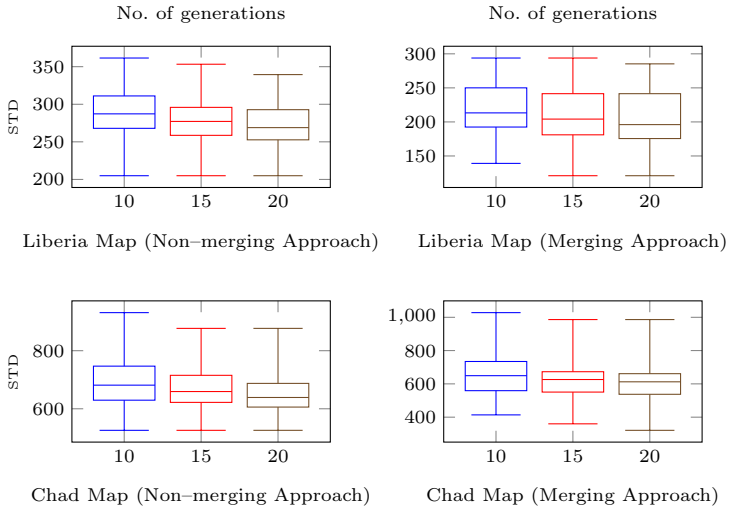


Fig. 14: Dataset 3, comparison between non-merging approach (left column) and merging approach (right column), No. of runs=51, pop size= 10

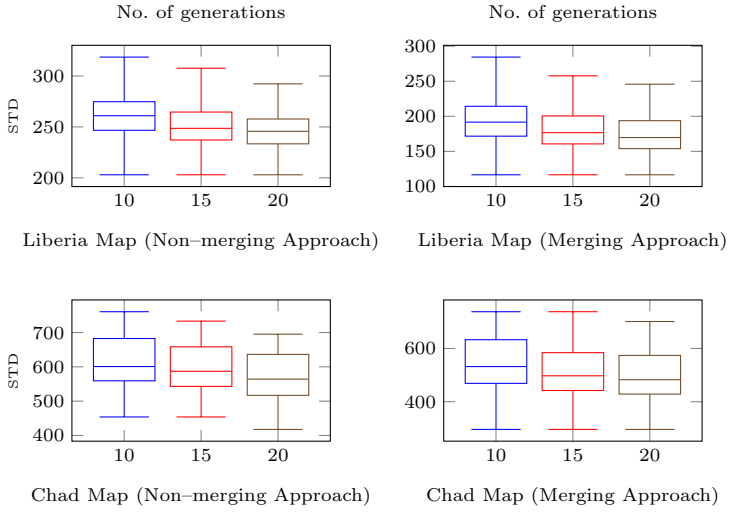


Fig. 15: Dataset 3, comparison between non-merging approach (left column) and merging approach (right column), No. of runs=51, pop size= 20

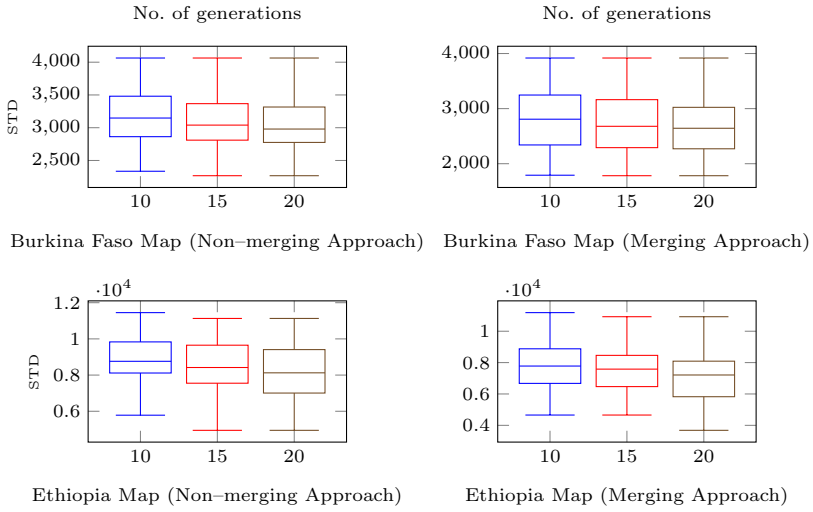


Fig. 16: Dataset 4, comparison between non-merging approach (left column) and merging approach (right column), No. of runs=51, pop size= 10

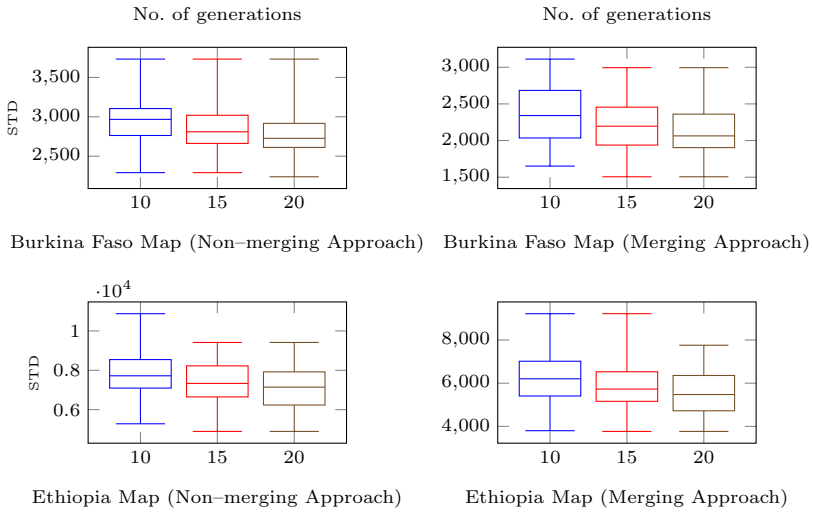


Fig. 17: Dataset 4, comparison between non-merging approach (left column) and merging approach (right column), No. of runs=51, pop size= 20

range is further reduced between 116.6 and 245.8; (top right in Figure 15. When comparing both experiments with population size of 20 and generation size of 20, it can be seen that the merging based experiment showed better standard deviation values than the non–merging based experiment. for example, for the map of Ethiopia (Figure 17, dataset 4), for the non-merging approach the standard deviation values range between 4895.7 and 9414.4 , while for the merging approach the range of the standard deviation values is between 3766.6 and 7757.7. All mentioned trends – i.e. the results improve with increasing population size, the results improve with increasing numbers of generations and the results improve in the merging approach compared with the non-merging approach – can be observed for all datasets: Dataset 1 (Figures 10 and 11), Dataset 2 (Figures 12 and 13), Dataset 3 (Figures 14 and 15) and Dataset 4 (Figures 16 and 17).

6 Conclusion and further research

In this paper, we discussed and highlighted the importance of spatial workload balancing for GIS vector map partitioning. To address this problem, we proposed an evolutionary-based approach for GIS map partitioning using the Genetic Algorithm (GA). Our proposed approach considers the nature of spatial data to increase the computation performance for processing GIS polygon-based maps with massive number of vertices and complex shapes. Four datasets were used, where each dataset had varying degrees of size in terms of number of polygons and number of vertices. Each dataset contained two maps, which had opposite ratios of number of vertices per polygon. A set of experiments on the four datasets were implemented to assess the influence of the evolutionary genetic algorithm parameters including the population size and the number of generations. The results showed the advantage of the proposed spatial workload balancing approach in real GIS map based partitioning scenarios. The use of evolutionary computation shows a promising potential in partitioning GIS maps into spatially balanced set of adjacent polygons based on the number of vertices.

The proposed approach in this paper is a first step toward a developed spatial workload balancing approach for GIS vector maps. Further research and experiments will be carried out on addressing the problem of the randomness in the map polygon shapes (concave and convex shapes) to further understand the behavior of the spatial workload balancing approach in extreme cases. Also, the possibility of introducing different weights for the different topological aspects will be investigated.

References

1. GIS (Geographic Information System) Overview. https://www.esri.com/en-us/what-is-gis/overview
2. Aji, A., Wang, F., Vo, H., Lee, R., Liu, Q., Zhang, X., Saltz, J.: Hadoop-GIS: A high performance spatial data warehousing system over mapreduce. In: The 39th International Conference on Very Large Data Bases, vol. 6, pp. 1009 – 1020 (2013)

3. Araujo Neto, A.C., Coelho da Silva, T.L., de Farias, V.A.E., Macêdo, J.A.F., de Castro Machado, J.: G2P: A partitioning approach for processing dbscan with mapreduce. In: Web and Wireless Geographical Information Systems, pp. 191–202. Springer International Publishing, Cham (2015)
4. Bação, F., Lobo, V., Painho, M.: Applying genetic algorithms to zone design. Soft Computing **9**(5), 341–348 (2005)
5. Barua, H.B., kumar Das, D., Sarmah, S.: A density based clustering technique for large spatial data using polygon approach. Journal of Computer Engineering **3**(6), 1–9 (2012)
6. Boobalan, M.P., Lopez, D., Gao, X.: Graph clustering using k-neighbourhood attribute structural similarity. Applied Soft Computing **47**, 216 – 223 (2016)
7. Cao, Z., Wang, S., Forestier, G., Puissant, A., Eick, C.F.: Analyzing the composition of cities using spatial clustering. In: Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing, pp. 141–148 (2013)
8. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. In: the 6th conference on Symposium on Opearting Systems Design and Implementation, pp. 1–13. Google, Inc. (2004)
9. Eldawy, A., Alarabi, L., Mokbel, M.F.: Spatial partitioning techniques in SpatialHadoop. Proc. VLDB Endow. **8**(12), 1602–1605 (2015)
10. Eldawy, A., Mokbel, M.F.: SpatialHadoop: A MapReduce framework for spatial data. In: The 31st International Conference on Data Engineering, pp. 1352–1363 (2015)
11. Ericsson, A., WCDMA, R.: Clustering and polygon merging algorithms for fingerprinting positioning in lte. In: 5th International Conference on Signal Processing and Communication Systems (ICSPCS), pp. 1–10 (2011)
12. ESRI: Esri shapefile technical description. Tech. rep., Environmental Systems Research Institute, Inc., 380 New York Street, Redlands, CA 92373-8100 USA (1998)
13. Fu, Y.X., Zhao, W.Z., Ma, H.F.: Research on parallel dbscan algorithm design based on mapreduce. In: Advanced Measurement and Test, *Advanced Materials Research*, vol. 301, pp. 1133–1138. Trans Tech Publications (2011)
14. Gu, X., Angelov, P.P., Príncipe, J.C.: A method for autonomous data partitioning. Information Sciences **460-461**, 65 – 82 (2018)
15. Guest, O., Kanayet, F., Love, B.: Gerrymandering and computational redistricting. Journal of Computational Social Science **2**, 119–131 (2019)
16. Gufler, B., Augsten, N., Reiser, A., Kemper, A.: The Partition Cost Model for Load Balancing in MapReduce, chap. 5, pp. 371–387. Springer New York, New York, NY (2012)
17. Jasim, M., Asadi, T.A.: New graph mining algorithm for vector GIS systems. In: 8th International Conference on Computing Technology and Information Management (NCM and ICNIT), vol. 1, pp. 335–338 (2012)
18. Ji, G., Zhang, L.: A spatial polygon objects clustering algorithm based on topological relations for GML data. In: 2009 International Conference on Information Engineering and Computer Science, pp. 1–4 (2009)
19. Joshi, D., Samal, A., Soh, L.K.: A dissimilarity function for clustering geospatial polygons. In: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 384–387. ACM, New York, NY, USA (2009)
20. Joshi, D., Samal, A.K., Soh, L.K.: Density-based clustering of polygons. In: 2009 IEEE Symposium on Computational Intelligence and Data Mining, pp. 171–178 (2009)

21. Joshi, D., Soh, L.K., Samal, A.: Redistricting using heuristic-based polygonal clustering. In: 2009 Ninth IEEE International Conference on Data Mining, pp. 830–835 (2009)
22. Joshi, D., Soh, L.K., Samal, A.: Redistricting using constrained polygonal clustering. IEEE Transactions on Knowledge and Data Engineering **24**(11), 2065 – 2079 (2012)
23. Kisore, N.R., Koteswaraiah, C.B.: Improving atm coverage area using density based clustering algorithm and voronoi diagrams. Information Sciences **376**, 1 – 20 (2017)
24. Levin, H.A., Friedler, S.A.: Automated congressional redistricting. Journal of Experimental Algorithmics **24**, 1–24 (2019)
25. Li, X., Li, W., Anselin, L., Rey, S., Koschinsky, J.: A MapReduce algorithm to create contiguity weights for spatial analysis of big data. In: Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data, pp. 50–53. ACM, New York, NY, USA (2014)
26. Liu, R., Wang, H., Yu, X.: Shared-nearest-neighbor-based clustering by fast search and find of density peaks. Information Sciences **450**, 200 – 226 (2018)
27. Longley, P.A., Goodchild, M., Maguire, D.J., Rhind, D.W.: Geographic Information Systems and Science, 3 edn. John Wiley and Sons (2011)
28. Photis, Y.N.: Redefinition of the greek electoral districts through the application of a region-building algorithm. MPRA Paper 42398, University Library of Munich, Germany (2012)
29. Puri, S., Agarwal, D., He, X., Prasad, S.K.: Mapreduce algorithms for gis polygonal overlay processing. In: 2013 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum, pp. 1009–1016 (2013)
30. Qiu, Q., Yao, X., Chen, C., Liu, Y., Fang, J.: A spatial data partitioning and merging method for parallel vector spatial analysis. In: 2015 23rd International Conference on Geoinformatics, pp. 1–5 (2015)
31. Schutzman, Z.: Trade-offs in fair redistricting. In: AAAI/ACM Conference on AI, Ethics, and Society, p. 159–165 (2020)
32. Shuliang, W., Gangyi, D., Ming, Z.: Big spatial data mining. In: IEEE International Conference on Big Data, pp. 13–21 (2013)
33. Wang, S., Chen, C.S., Rinsurongkawong, V., Akdag, F., Eick, C.F.: A polygon-based methodology for mining related spatial datasets. In: Proceedings of the 1st ACM SIGSPATIAL International Workshop on Data Mining for Geoinformatics, pp. 1–8 (2010)
34. Wang, S., Eick, C.F.: A polygon-based clustering and analysis framework for mining spatial datasets. GeoInformatica **18**(3), 569–594 (2014)
35. Wang, W., Du, S., Guo, Z., Luo, L.: Polygonal clustering analysis using multilevel graph-partition. Transactions in GIS **19**(5), 716–736 (2015)
36. Wei, H., Du, Y., Liang, F., Zhou, C., Liu, Z., Yi, J., Xu, K., Wu, D.: A k-d tree-based algorithm to parallelize kriging interpolation of big spatial data. GIScience & Remote Sensing **52**(1), 40–57 (2015)
37. Zhang, J., Samal, A., Soh, L.: Polygon-based spatial clustering. In: The 8th International Conference on GeoComputation, pp. 1–5 (2005)
38. Zhang, X., Huang, B., Tay, R.: Estimating spatial logistic model: A deterministic approach or a heuristic approach? Information Sciences **330**, 358 – 369 (2016). SI Visual Info Communication
39. Zhao, L., Chen, L., Ranjan, R., Choo, K.K.R., He, J.: Geographical information system parallelization for spatial big data processing: a review. Cluster Computing **19**(1), 139–152 (2016)