Performance Analysis of Isoperimetric Algorithm Applied to CT Images

Priyanka Kumar
Electronics and Telecome. Engg
Maharashtra Institute of Technology,
Pune, India

Dr. Arti Khaparde
Electronics and Telecome. Engg
Maharashtra Institute of Technology,
Pune, India

***Abstract*— Isoperimetric algorithm is one of the effective algorithm which solves the problem of Graph Partitioning and is based on the optimization of Isoperimetric constant. Isoperimetric algorithm is easy to parallize and works smoothly on non-planar graphs and has wide applications in image processing fields such as Image Segmentation. This paper is intented to analyze the various parameters of the above algorithm e.g. scale, stop, cutoff, connectivity and recursion cap for segmentation purposes when applied to CT images.**

***Index Terms*—Isoperimetric constant, Graph Partitioning, Image Segmentation, Isoperimetric algorithm.**

I. INTRODUCTION

Image Segmentation is one of the most important area in the field of Image Processing, which means dividing the image into its constituent regions or objects. One of the most difficult problems in segmentation is to find the region of minimum area so that the smallest part could be obtained from the entire image. Out of the many algorithms which have been proposed to solve the above problem, isoperimetric algorithm is better as it solves the problem of Graph Partitioning and gives us the optimized segments.

A. Graph Partitioning Problem

The Graph Partitioning problem is to choose the partition from the vertex set such that the sum of the edge weights in the entire vertex set is about the same. The chosen partition should share the minimal number of spanning edges while satisfying a constant known as the cardinality constraint.

Suppose given a graph such that G_r= (V, E), where V denotes the set of vertices and E denotes the set of edges, that determines the connectivity between the nodes, then the graph partitioning problem is to divide the graph G_r into n disjoint partitions such that the number of cuts in the edges of the cuts are minimized and also on the other hand, the weights of the subdomains should get reduced. The weight of subdomains means the algebraic sum of vertex weights allocated in it. Here both vertex and edges can be weighted so as to satisfy the cardinality constraint, where |v| denotes the cardinality constraint for weight of vertex v and |e| denotes the cardinality constraint for weight of edge e.

Graph Partitioning appears in many fields such as parallel processing, solving linear systems of sparse type for VLSI circuit design and image segmentation.

II. ISOPERIMETRIC ALGORITHM

A. The classic isoperimetric problem [5]

The Graph partitioning has been strongly influenced by the classic isoperimetric problem, which is to find the boundary of minimum perimeter enclosing maximal area and the above isoperimetric problem totally depends upon the isoperimetric constant h_i. Defining the isoperimetric constant h_i for a manifold as:

$$h_i = \inf_s \frac{|\partial S|}{Vs} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{(1)}$$

where S is the region in the manifold, |∂S| is the area of the region of boundary S, V_S denotes the volume of region S and h_i is the infimum of ratio over all possible S. If V_T denotes the total volume, than for a compact manifold $Vs \leq 0.5\ V_T$ and for a non compact manifold Vs is less than infinity.

B. The Isoperimetric algorithm [5]

A graph is a pair G_r = (V, E) with vertices (nodes) v ∈ V and edges e ∈ E ⊆ V X V. An edge, e, spanning two vertices, v_i and v_j, is denoted by e_{ij}. A weighted graph is more general than unweighted graph hence we have developed our results for weighted graphs. Here a weighted graph typically is assigned a real and non negative value called a weight and the weight of the edge is denoted by w (e_{ij}) or w_{ij}. The degree of the vertex v_i denoted by d_i is given by,

$$d_i = \sum_{e_{ij}} w_{ij} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{(2)}$$

For a graph G_r the isoperimetric constant is defined in “Eq.1”. Since we are dealing with finite graphs, so in graphs with finite node set the infimum in “Eq.1” becomes minimum.

Hence we will use the word minimum in place of infimum. So "E.q.1" can now be written as,

$$h_i = \min \frac{|\partial S|}{Vs} \quad \ldots\ldots\ldots\ldots (3)$$

The boundary of a set S is now defined as $\partial S = \{e_{ij} \mid i \in S, j \in \bar{S}\}$, $\bar{S}$ denotes the set complement and

$$|\partial S| = \sum_{e_{ij} \in \partial S} w_{ij} \quad \ldots\ldots\ldots\ldots (4)$$

For a given set S, we term the isoperimetric ratio as h(S). The segmentation is better if the isoperimetric ratio is low and this is possible if $h(S) = h_i$, where S and $\bar{S}$ could be considered as the isoperimetric sets for the graph G_r, which could be called as the heuristic sets with lowest isoperimetric ratio for the purpose of partition that runs in low order polynomial time.

C. Derivation of Isoperimetric algorithm [5]

Define an indicator vector y that takes binary values at each node,

$$y_i = 1 \text{ , if } v_i \in S \ldots\ldots\ldots\ldots (5)$$

$$y_i = 0 \text{ , otherwise} \ldots\ldots\ldots\ldots (6)$$

Define a pxp matrix L, also known as Laplacian matrix in the context of finite difference method and the admittance matrix in the context of circuit theory as,

$$L_{vi\,vj} = d_i \qquad if\ i = j \ldots\ldots\ldots\ldots (7)$$

$$L_{vi\,vj} = -w(e_{ij}) \qquad if\ e_{ij} \in E \ldots\ldots (8)$$

$$L_{vi\,vj} = 0 \text{ , otherwise} \quad \ldots\ldots\ldots\ldots (9)$$

By the definition of Laplacian matrix,

$$|\partial S| = y^T L y \ldots\ldots\ldots\ldots (10)$$

If k indicates the vector of all ones, then the volume S can be maximized subject to the condition

$$V_s \leq 0.5\, V_T = 0.5\, k^T d \ldots\ldots\ldots\ldots (11)$$

may be done by asserting the constraint as,

$$y^T d = 0.5 k^T d \ldots\ldots\ldots\ldots (12)$$

Thus, the isoperimetric constant of a graph may be written in terms of indicator vector y as,

$$h_i = \min_y \frac{y^T L y}{y^T d} \ldots\ldots\ldots\ldots (13)$$

subject to the condition as specified in "Eq.11". For the indicator vector y, h(y) denotes the isoperimetric ratio associated with the partition y. The constrained optimization of the isoperimetric ratio can be obtained by the introduction of Lagrange multiplier to take the positive real values by minimizing the cost function given by the equation,

$$Q(y) = y^T L y - \Lambda (y^T d - 0.5\, k^T d) \ldots\ldots (14)$$

now differentiating "Eq.14" w.r.t y will yield a minimal partition so the differentiation gives us,

$$\frac{dQ(y)}{dy} = 2Ly - \Lambda d \quad \ldots\ldots\ldots\ldots (15)$$

So, the problem of finding minimum partition gets reduced to solving the linear system as,

$$2Ly = \Lambda d \ldots\ldots\ldots\ldots (16)$$

Since, we are interested with only the unique solution of "Eq.16", the scalar multiplier Λ is ignored but as matrix L is singular, finding the unique solution to the above equation requires additional constraint, which could be done by choosing the node of largest degree as the ground node vg, and determining L_0 and d_0 by eliminating the row/column corresponding to vg such that,

$$L_0 y_0 = d_0 \ldots\ldots\ldots\ldots (17)$$

which then becomes the nonsingular system of equations. Solving the "Eq.17" will then yield a real valued solution which may be considered as a partition by setting the threshold. This algorithm can be applied recursively to each partition separately until the condition for obtaining the lowest isoperimetric ratio is obtained. Lower the isoperimetric ratio, better is the partition and if the isoperimetric ratio fails to meet the predetermined threshold, then we term this threshold as the stop parameter which should be in the interval (0,1).The isoperimetric algorithm considers a condition of finding a lowest isoperimetric ratio which runs in low order polynomial time.

III. IMPLEMENTATION

The flowchart for the implementation of isoperimetric algorithm is shown in Fig (1).

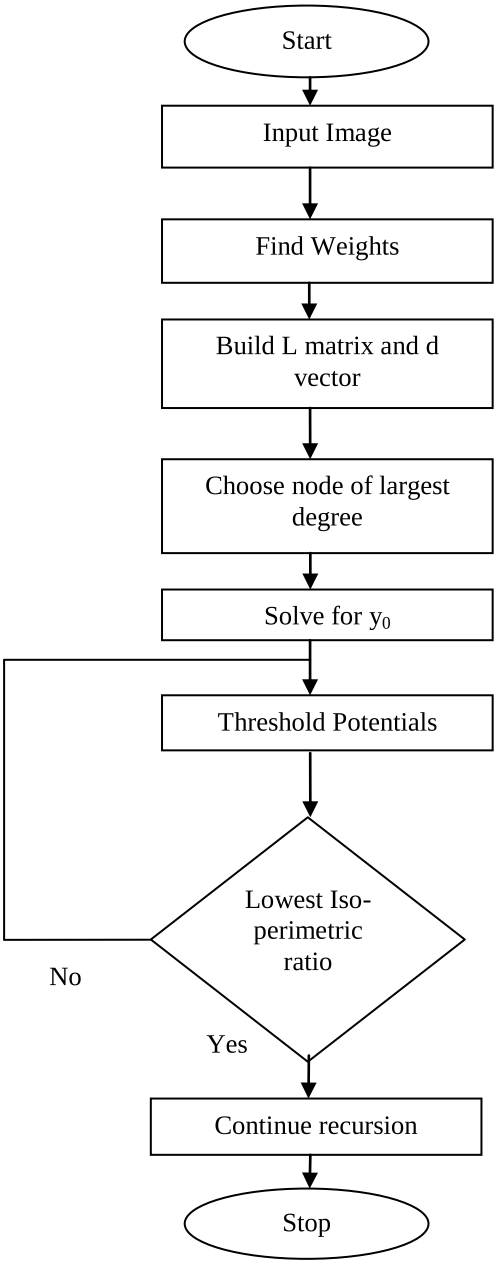

Fig1.Flowchart of isoperimetric algorithm for image segmentation.

The implementation of isoperimetric algorithm for image segmentation can be summarized in the followings steps as:

- Find weights for all edges.
- Build the L matrix and d vector using the "equations 7, 8 and 9".
- Choose the ground node vg as the node of largest degree, and determine L_0 and d_0 by eliminating the row/column corresponding to ground node.
- Solve "Eq.17" for y_0.
- Threshold the potentials y at the value till the lowest isoperimetric ratio is obtained.
- Continue the recursion until the isoperimetric ratio is larger than the stop parameter.

IV. RESULTS

The isoperimetric algorithm was applied to the different CT images so as to analyze the various input parameters. The different set of values for each input parameter e.g ValScale, stop, cutoff, recursion cap and connectivity was applied to analyze the various segments. The average time of each parameter was also computed to prove that the above algorithm runs in low order polynomial time. The below section from A to E gives the subjective analysis while the section F gives computational time analysis.

A. Val Scale: This parameter is used for generation of weights. The segmentation efficiency depends upon the type of image. The results of above parameter when applied to the CT image for the ValScale values are shown in Fig (2), Fig (3) and Fig (4).

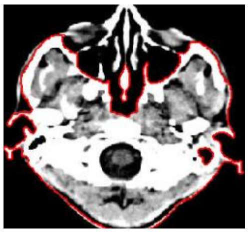

Fig 2. Segmentation for ValScale=50

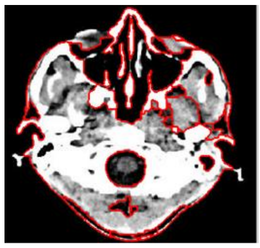

Fig 3. Segmentation for ValScale=110

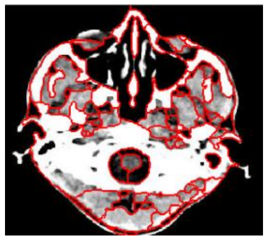

Fig 4. Segmentation for ValScale=200

From the above results, it is seen that for this image 110 is better Valscale value as it gives the optimized segmentation. For the valscale value 50, the image is under segmented , while for the valscale value 200, the image is over segmented.

B. Stop: This parameter provides the maximum allowable isoperimetric ratio above which the execution of the

ISSN: 2278-0181
Vol. 3 Issue 6, June - 2014

isoperimetric algorithm stops. The results of above parameter for the stop values 1e-5, 1e-4 and 1e-3 are shown in Fig (5), Fig (6) and Fig (7).

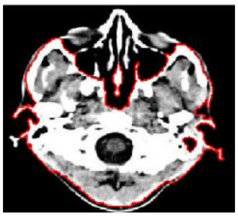

Fig 5. Segmentation for Stop=1e-5

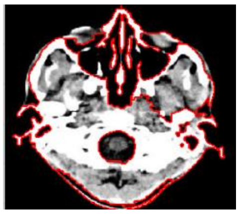

Fig 6. Segmentation for Stop=1e-4

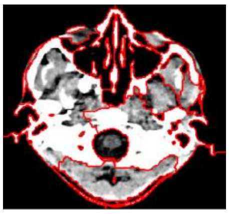

Fig 7. Segmentation for Stop=1e-3

For this image 1e-4 is better option.

C. Cut off: This parameter gives the minimum number of nodes that can be considered for a valid segment. We have chosen the three values 40,100 and 900 which could be considered as the minimum, medium and maximum value. The results of above parameter are shown in Fig (8), Fig (9) and Fig (10).

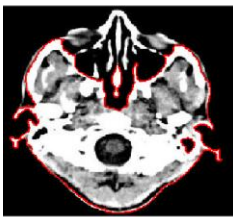

Fig 8. Segmentation for Cut off=40

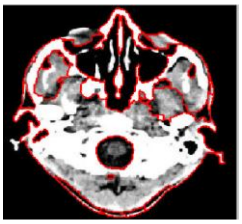

Fig 9. Segmentation for Cut off=100

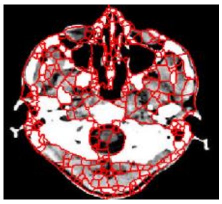

Fig 10. Segmentation for Cut off=900

From the above results it can be seen that 100 is better cutoff value.

D. Recursion Cap: This parameter gives the number of iterations so as to provide optimized segmentation. The results of above parameter for the values 5, 9 and 90 are shown in Fig (11), Fig (12) and Fig (13).

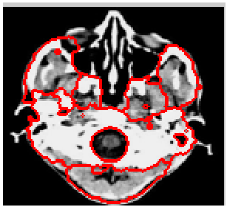

Fig 11. Segmentation for Recursion Cap= 5

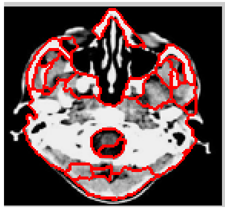

Fig 12. Segmentation for Recursion Cap= 9

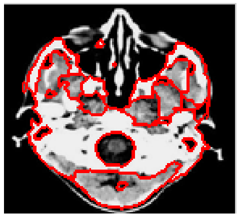

Fig 13. Segmentation for Recursion Cap= 90

From the above results it can be seen that 5 is better value for recursion cap.

E. Connectivity: This parameter is used for comparing neighbouring pixels on connectivity basis. Many connectivity schemes can be used, but here we have used two connectivity schemes namely 4-point Connectivity and 8-point Connectivity.

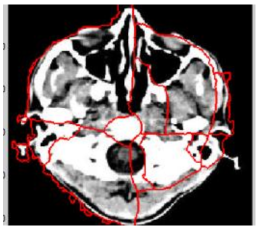

Fig 14. Segmentation using 4-point Connectivity Scheme.

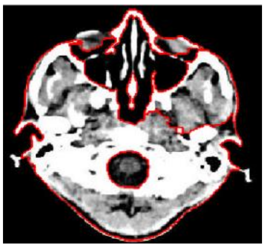

Fig 15. Segmentation using 8-point Connectivity Scheme.

It is seen that 8-point connectivity is better as it avoids over segmentation.

F. Computational Time Analysis

The average time of each parameter was computed w.r.t to other parameters to analyze their performances, that the optimized Segmentation is obtained in less time. The results for each parameter are shown in Table(Ia), Table(Ib), Table(IIa), Table(IIb) ,Table(IIIa), Table(IIIb), Table(IVa) and Table(IVb).

Table I(a): Average time for valscale parameter using 4-point connectivity scheme:

Parameter	***Value***		
	min	***med***	***max***
Val Scale	50	110	200
Stop	1e-5	1e-5	1e-5
Cut off	40	40	40
Recursion Cap	5	5	5
Connectivity	4-point	4-point	4-point
Time(in seconds)	3.004	3.35	3.69

Table I(b): Average time for valscale parameter using 8-point connectivity scheme:

Parameter	***Value***		
	min	***med***	***max***
Val Scale	50	110	200
Stop	1e-5	1e-5	1e-5
Cut off	40	40	40
Recursion Cap	5	5	5
Connectivity	8-point	8-point	8-point
Time(in seconds)	2.9102	3.250	3.60

Table II(a): Average time for stop parameter using 4-point connectivity scheme:

Parameter	***Value***		
	min	***med***	***max***
Val Scale	50	50	50
Stop	1e-5	1e-4	1e-3
Cut off	40	40	40
Recursion Cap	5	5	5
Connectivity	4-point	4-point	4-point
Time(in seconds)	3.00	3.58	3.70

Table II(b): Average time for stop parameter using 8-point connectivity scheme:

Parameter	***Value***		
	min	***med***	***max***
Val Scale	50	50	50
Stop	1e-5	1e-4	1e-3
Cut off	40	40	40
Recursion Cap	5	5	5
Connectivity	8-point	8-point	8-point
Time(in seconds)	2.88	3.43	3.57

Table III(a): Average time for cut off parameter using 4-point connectivity scheme:

Parameter	***Value***		
	min	***med***	***max***
Val Scale	50	50	50
Stop	1e-5	1e-5	1e-5
Cut off	40	100	900
Recursion Cap	5	5	5
Connectivity	4-point	4-point	4-point
Time(in seconds)	2.92	3.00	3.03

Table III(b): Average time for cut off parameter using 8-point connectivity scheme:

Parameter	***Value***		
	min	***med***	***max***
Val Scale	50	50	50
Stop	1e-5	1e-5	1e-5
Cut off	40	100	900
Recursion Cap	5	5	5
Connectivity	8-point	8-point	8-point
Time(in seconds)	2.89	2.97	3.002

Table IV(a): Average time for recursion cap parameter using 4-point connectivity scheme:

Parameter	***Value***		
	min	***med***	***max***
Val Scale	50	50	50
Stop	1e-5	1e-5	1e-5
Cut off	40	40	40
Recursion Cap	5	9	90
Connectivity	4-point	4-point	4-point
Time(in seconds)	2.92	2.86	2.99

Table IV(b): Average time for recursion cap parameter using 8-point connectivity scheme:

Parameter	***Value***		
	min	***med***	***max***
Val Scale	50	50	50
Stop	1e-5	1e-5	1e-5
Cut off	40	40	40
Recursion Cap	5	9	90
Connectivity	8-point	8-point	8-point
Time(in seconds)	2.89	2.90	2.96

V. CONCLUSION

Isoperimetric algorithm is one of the effective algorithm, which solves the problem of graph partitioning. From the results it can be concluded that the isoperimetric algorithm is better for image segmentation as it runs in low order polynomial time. To get the optimized segmentation in less time the valscale parameter and stop parameter could be considered high while the cutoff parameter and the recursion cap could be considered low with 8-point connectivity scheme.

REFERENCES

[1] A. Daneshgar, R. Javadi, S.B. Shariat Razavi, "**Clustering and outlier detection using isoperimetric number of trees,**", *IEEE Transactions on Pattern Recognition, Volume 46, Issue 12, Pages 3371- 3382, December 2013.*

[2] Arti Khaparde, J.Saritha, R.R Vishnuvardhan, ***"Image Segmentation using Isoperimetric algorithm***", *10th WSEAS International conference on Recent Researches in Telecommunications, Informatics, Electronics and Signal Processing*, Lanzarote Canary Islands, Spain, May 27-29, 2011.

[3] Daming Zhang ; Maosheng Fu ; Dengdi Sun ; Bin Luo,,"**Image threshold selection with Isoperimetric partition**," *5th International Conference on Computer Science and Education (ICCSE)*, vol. 10, pages 328 – 333, pp. 978-1-4244-6002-1, Aug 2010.

[4] Leo Grady and Eric L. Schwartz, "***Isoperimetric Graph Partitioning for Image Segmentation***," *IEEE Transactions on Analysis and Machine Intelligence*, vol. 28, No. 3, March 2006, pp. 469-475.

[5] Leo Grady and Eric L. Schwartz, "***Isoperimetric Graph Partitioning for Data Clustering and Image Segmentation***," *IEEE Transactions on Pattern Analysis and Machine Intelligence* vol. XX, No. Y, Month 2004.

[6] Graph Theory, image processing, **www.wikipedia.com/**