Volume 2, No. 8, August 2011
Journal of Global Research in Computer Science
ISSN-2229-371X
RESEARCH PAPER
Available Online at www.jgrcs.info

OVERVIEW OF GRAPH PARTITIONING WITH NATURAL HEURISTICS CUTS

Dilpreet Kaur[1], Balwinder Singh[2]
[1, 2] Sri Sukhmani Engg. And Technology, Yadwindera Engg. College
dilpreet410@gmail.com

***Abstract*—** This paper will present a short overview of several approaches to solve the k-way graph partitioning problem. In short this problem considers the partitioning of a graph in k partitions, in such a way that one minimizes the cut value. The cut value represents the number of edges crossing this partitions.The technique for simultaneous segmentation and classification of image partitions using graph cuts. By combining existing image segmentation approaches with simple learning techniques. We manage to include prior knowledge into this visual grouping process.

Keywords **-** graph partitioning; road networks; minimum cuts; maximum flows; algorithms, image segmentation.

INTRODUCTION

The graph partitioning problem consists of dividing a graph into equal sections, such that the cut separating these sections has the lowest cost of all possible partitions. The general graph partitioning problem is the k-way graph portioning: to create a partitioning of k equally sized partitions having minimal cut cost. Applications of graph partitioning are many, including VLSI circuit design, logistics, data mining, parallel computing and coloring. In this paper we shall consider the 2-way graph partitioning problem, but also show how to extend results to problems calling for more sections. Even in this relatively simple setting, the problem is too large to solve by mere enumeration of partitions, as the number of partitions is exponential in the number of nodes in the graph. Therefore many heuristics have been devised to solve the graph partitioning problem within a reasonable amount of time. A heuristic may yield a good solution quickly. As exhaustive methods will take very long to find a solution, speed is an important metric for comparing different heuristics.

However, no heuristic may guarantee the result will be optimal: the solution method can get stuck in local optimum. This paper will therefore discuss a new algorithm for finding solutions to the graph partitioning problem. The main question we will address in this paper, is how well and to what extend several local search heuristics perform against each other. These approaches include: Genetic Algorithm (GA), Kernighan-Lin (KL), GA+KL, Our gym-class heuristic (GC) and GC+KL. To compare these five approaches, the main criteria will be the time complexity, but mainly empirical time performance, and the minimum cut value found. We will take the Kernighan-Lin heuristic as a benchmark, as it's good performance has been established for already a few decades . We will show, by fitting each of the algorithms into the local search template of Aarts, Lenstra and Vaessens [1], Genetic Algorithms belong to the class of local search algorithms.

THEORY OF GRAPH CUTS

A graph cut is the process of partitioning a directed or undirected graph into disjoint sets. The concept of optimality of such cuts is usually introduced by associating energy to each cut. Problems of this kind have been well studied within the field of graph theory but can for graphs with more than only a few nodes be notoriously difficult. Nevertheless, ever since it became apparent that many low-level vision problems can be posed as finding energy minimizing cuts in graphs these techniques have received a lot of attention in the computer vision community. Graph cut methods have been successfully applied to stereo, image restoration, and texture synthesis and image segmentation. Below we give a brief overview of graph cuts for image segmentation as well as an introduction to some basic definitions.

Min-cut/Max-flow cuts:

Given a graph **G = {V, E, W}** , where **V** denotes its nodes, **E** its edges and **W** the affinity matrix, which associates a weight to each edge in E. A cut on a graph is a partition of V into two subsets A and B. Perhaps the simplest and best known graph cut method is the min-cut formulation. The min-cut of a graph is the cut that partitions G into disjoints segments such that the sum of the weights associated with edges between the different segments is minimized.

However, as this is an NP-hard combinatorial optimization problem, the task of finding the solution can be a formidable one. In order to overcome this one can relax (1) into a semi-definite program, resulting in a convex problem for which efficient solvers exist. However, the task of finding the solution to the original problem from the relaxed one still remains an open issue. Another commonly used approach is based on a slight reformulation of the original min-cut problem. By adding the requirement that two predefined nodes, denoted terminal nodes or source and sink nodes, in G must be in separate sets, the complexity of the problem is significantly reduced. Finding the min-cut separating the source and the sink, the s-t cut, can be achieved in polynomial time. If one views the weights associated to each node as a flow capacity it can be shown that the maximal amount of a flow from source to sink is equal to the capacity of a minimal cut. Therefore the min-cut problem is also known as the max-flow problem.

The Image Seen as a Graph:

The general approach to constructing an undirected graph from an image is shown in fig 1

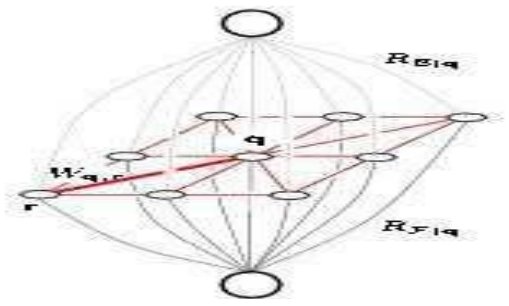

Figure 1:

Graph representing a 3-by-3 image. Basically each pixel in the image is viewed as a node in a graph, edges are formed between nodes with weights corresponding to how alike two pixels are, given some measure of similarity, as well as the distance between them. In attempt to reduce the number of edges in the graph only pixels within a smaller, predetermined neighborhood N of each other are considered.

The two terminal nodes, the source and the sink does not correspond to any pixel in the image but instead are viewed as representing the object and background respectively. Edges are formed between the source and sink and all other non-terminal nodes, where the corresponding weights are determined using models for the object and background. The min-cut of the resulting graph will then be the segmentation of the image at hand. This segmentation should then be a partition such that, flowing to the definition of image-pixel resemblance, similar pixels close to each other will belong to the same partition. In addition, as a result of the terminal weights, pixels should also be segmented in such a manner so they end up in the same partition as the terminal node corresponding to the model (object or background) they are most similar to an illustration of the segmentation process can be seen in figure 2.

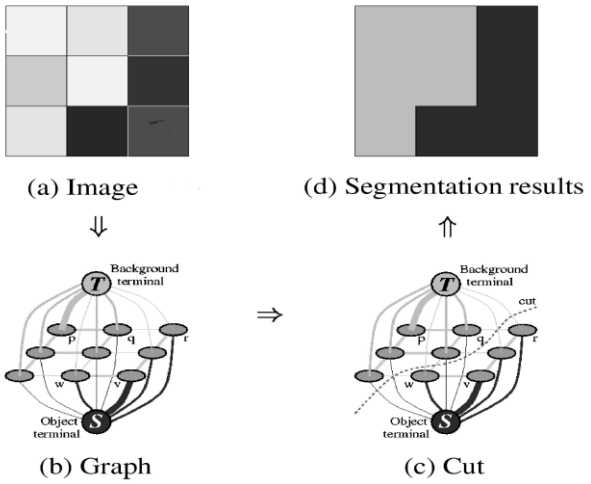

Figure 2: Example segmentation of a very simple 3-by-3 image. Edge thickness corresponds to the associated edge weight. (Image courtesy of Yuri Boykov)

EVALUATION AND COMPARISON OF IMAGE SEGMENTATION

The research on evaluation of image segmentation can provide crucial reference for those segmentation algorithms, and so this research deserves wide attentions. Understandably, the basic requirements are as follows: universal use for evaluation algorithms, its simplification and reliability, and whether referent images or manual intervention is needed. Generally, two basic methods are applied to objective evaluation of image segmentation: analytical technique and experimental technique.

The Analytical Technique:

The analytical technique evaluates an image segmentation algorithm by analyzing the principle of the algorithm, its complexity, the prior knowledge needed, accurate detecting probability, image resolution and so forth. The analytical technique usually provides supplementary information and supports for other methods of segmentation evaluation and it is seldom used alone.

The Experimental Technique:

The experimental technique, which is widely used, interprets and compares experiment results of image segmentation algorithms to make an evaluation. This technique can be subdivided into two distinct methods: superiority evaluation method and deviation evaluation method.

The Superiority Evaluation Method: The superiority evaluation method evaluates an image segmentation algorithm by utilizing human visual trait. It judges the quality of a segmentation algorithm by calculating certain measures based on image segmentation result. The commonly used measures are region uniformity, contrast of regions, region shape and synthetically measure based on ambiguity. The evaluation method based on region uniformity characterizes segmentation result by quantizing uniformity within regions after segmentation. Suppose *Ri* stands for region *i.*

The Deviation Evaluation Method: In this method, firstly a standard segmentation image is provided for comparison criteria. Then the disparity between actual segmentation and ideal one can be calculated to evaluate the image segmentation algorithm. With a comparing test, the deviation evaluation method is generally more effective than the superiority evaluation method. Generally, this method executes evaluation via factors as follows: the probability of mistaken pixels, the position of mistaken pixels, the consistency for the number of regions and so on. The evaluation method based on the consistency for the number of regions evaluates image segmentation in the manner like this: suppose that *N'* stands for the number of regions after image segmentation and *N* is the number of regions correctly partitioned. Reasonably, we can evaluate image segmentation algorithms by analyzing the difference between *N'* and *N*.

PARTITIONING USING NATURAL CUT HEURISTICS

PUNCH (Partitioning Using Natural Cut Heuristics), a partitioning algorithm tailored to graphs containing natural cuts, such as road networks. Given a parameter U (the maximum size of any cell), PUNCH partitions the graph into cells of size at most U while minimizing the number of edges between cells.The algorithm runs in two phases: filtering and assembly. The filtering phase aims to reduce the size of the graph significantly while preserving its overall structure. It keeps the edges that appear in natural cuts, relatively sparse cuts close to denser areas, and contracts other edges. The notion of natural cuts and efficient algorithms to compute them are the main

contributions of our work. Note that to find a natural cut it is not enough to pick a random pair of vertices and run a minimum cut computation between them: because the average degree in road networks is small, this is likely to yield a trivial cut. We do better by finding minimum cuts between carefully chosen regions of the graph. Edges that never contributed to a natural cut are contracted, potentially reducing the graph size by orders of magnitude. Although it is smaller, the filtered (contracted) graph preserves the natural cuts of the input.

The second phase of our algorithm (assembly) is the one that actually builds a partition. Since the filtered graph is much smaller than the input, we can use more powerful (and time-consuming) techniques in this phase. Another important contribution of our work is a better local search algorithm for the second phase. Note that the assembly phase only tries to combine fragments (the contracted regions). Unlike existing practitioners, we do not disassemble individual fragments. Note that we focus on finding partitions with small cells, but with no hard bound on the number of cells thus created. As already mentioned, previous work in this area has concentrated on finding balanced partitions, in which the total number of cells is bounded.

We show how one can use simple heuristics to transform the solutions found by our algorithm into balanced ones. Our comparison shows that PUNCH significantly improves the best previous bounds for road networks. We are not aware of any approach using min-cut computations to reduce the graph size in the context of graph partitioning. However, work on improving a partition is vast. For example, many of the algorithms within the MGP framework use local search based on vertex swapping, which improves the cut size by moving vertices from one cell to another. The most important ones are the FM and KL heuristics. The FM heuristic runs in worst-case linear time by allowing each vertex to be moved at most once. Local improvements based on minimum cuts often yield better results than greedy methods. For example, Andersen and Lang run several minimum cut computations to improve the cut between two neighboring cells. Another common approach to optimize a cut between two cells is based on parametric minimum cut computation. Besides vertex swapping and minimum cuts, local search based on diffusion gives good results as well. This approach has the nice side effect of optimizing the shape of the cells, but it requires an embedding of the graph. Most other methods, including ours, do not.

CONCLUSIONS

In this paper we have suggested a method for automatic detection, segmentation and classification of textured regions in color images. It describes how prior information can be brought into a graph cut framework through the use of terminal node weights and learning techniques. An efficient implementation is also presented along with some very promising results on an underwater image of a coral reef as well as an ordinary holiday picture. PUNCH, a new algorithm for graph partitioning that works particularly well on road networks. The key feature of PUNCH is its graph reduction routine: By identifying natural cuts and contracting dense regions, it can reduce the input size by orders of magnitude, while preserving the natural structure of the graph. Because of this efficient reduction in size, we can run more time-consuming routines to assemble a good partition. As a result, we obtain the best known partitions for road networks, improving previous bounds by more than 10% on average. Altogether, PUNCH is slower compared to some previous graph partitioning algorithms, but it needs only a few minutes to generate an excellent partition, which is fast enough for most applications.

REFERENCES

[1]. M. R. Garey and D. S. Johnson, Computers and Intractability. A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, 1979.

[2]. C. M. Fiduccia and R. M. Mattheyses, “A linear-time heuristic for improving network partitions,” in Proceedings of the 19th ACM/IEEE Conference on Design Automation, 1982, pp. 175–181

[3]. B. W. Kernighan and S. Lin, “An Efficient Heuristic Procedure for Partitioning Graphs,” Bell System Technical Journal, vol. 49, no. 2, pp. 291–307, February 1970.

[4]. R. Andersen and K. J. Lang, “An Algorithm for Improving Graph Partitions,” in Proceedings of the 19th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA’08), 2008, pp. 651–660

[5]. D. Pritchard and R. Thurimella, “Fast computation of small cuts via cycle space sampling,” ACM Transaction on Algorithms, 2010, to appear.

[6]. A. V. Goldberg and R. E. Tarjan, “A New Approach to the Maximum Flow Problem,” J. Assoc. Comput. Mach., vol. 35, pp. 921–940, 1988.

[7]. C. C. Ribeiro, E. Uchoa, and R. F. Werneck, “A hybrid GRASP with perturbations for the Steiner problem in graphs,” INFORMS J. Computing, vol. 14, no. 3, pp. 228–246, 2002.

[8]. Z. Wu and R. Leahy, “An Optimal Graph Theoretic Approach to Data Clustering: Theory and Its Application to Image Segmentation,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 15, no. 11, pp. 1101-1113, Nov. 1993.

[9]. I.J. Cox, S.B. Rao, and Y. Zhong, “Ratio Regions: A Technique for Image Segmentation,” Proc. Int’l Conf. Pattern Recognition, pp. 557- 564, Aug. 1996.

[10]. J. Shi and J. Malik, “Normalized Cuts and Image Segmentation,” Proc. IEEE CS Conf. Computer Vision and Pattern Recognition, pp. 731-737, 1997.

[11]. I.H. Jermyn and H. Ishikawa, “Globally Optimal Regions and Boundaries,” Proc. Seventh Int’l Conf. Computer Vision, pp. 904-910, 1999.

[12]. S. Sarkar and P. Soundararajan, “Supervised Learning of Large Perceptual Organization: Graph Spectral Partitioning and Learning Automata,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 22, no. 5, pp. 504-525, May 2000.

