Multilevel Spectral Clustering: Graph Partitions and Image Segmentation

by

Tian Fook Kong

B.Eng, Nanyang Technological University, Singapore (2007)

Submitted to the School of Engineering
in partial fulfillment of the requirements for the degree of
Master of Science in Computation for Design and Optimization

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2008

© Massachusetts Institute of Technology 2008. All rights reserved.

Author

School of Engineering
Aug 15, 2008

Certified by

Gilbert Strang
Professor of Mathematics
Thesis Supervisor

Accepted by

Jaime Peraire
Professor of Aeronautics and Astronautics
Codirector, Computation for Design and Optimization Program
Multilevel Spectral Clustering: Graph Partitions and Image Segmentation

by

Tian Fook Kong

Submitted to the School of Engineering
on Aug 15, 2008, in partial fulfillment of the requirements for the degree of Master of Science in Computation for Design and Optimization

Abstract

While the spectral graph partitioning method gives high quality segmentation, segmenting large graphs by the spectral method is computationally expensive. Numerous multilevel graph partitioning algorithms are proposed to reduce the segmentation time for the spectral partition of large graphs. However, the greedy local refinement used in these multilevel schemes has the tendency of trapping the partition in poor local minima.

In this thesis, I develop a multilevel graph partitioning algorithm that incorporates the inverse powering method with greedy local refinement. The combination of the inverse powering method with greedy local refinement ensures that the partition quality of the multilevel method is as good as, if not better than, segmenting the large graph by the spectral method. In addition, I present a scheme to construct the adjacency matrix, $W$ and degree matrix, $D$ for the coarse graphs.

The proposed multilevel graph partitioning algorithm is able to bisect a graph ($k = 2$) with significantly shorter time than segmenting the original graph without the multilevel implementation, and at the same time achieving the same normalized cut ($Ncut$) value. The starting eigenvector, obtained by solving a generalized eigenvalue problem on the coarsest graph, is close to the Fiedler vector of the original graph. Hence, the inverse iteration needs only a few iterations to converge the starting vector. In the $k$-way multilevel graph partition, the larger the graph, the greater the reduction in the time needed for segmenting the graph.

For the multilevel image segmentation, the multilevel scheme is able to give better segmentation than segmenting the original image. The multilevel scheme has higher success of preserving the salient part of an object. In this work, I also show that the $Ncut$ value is not the ultimate yardstick for the segmentation quality of an image. Finding a partition that has lower $Ncut$ value does not necessarily mean better segmentation quality. Segmenting large images by the multilevel method offers both speed and quality.

Thesis Supervisor: Gilbert Strang
Title: Professor of Mathematics
Acknowledgments

I am deeply indebted to my supervisor, Professor Gilbert Strang for his tutelage and advice in guiding me towards completing my Master's degree thesis. He has been very patient in explaining the concepts and approaches to me. He also constantly inspires and encourages me. Thank you for sharing your experiences with me.

In addition, I also owe a lot to the Singapore-MIT alliance program. I must thank Dr. John Desforge and Jocelyn Sales from the SMA office for making my stay at MIT enjoyable and memorable. The trips and gathering you guys put together for us were awesome. I have enjoyed every bit of time we have had together.

I would like to take this opportunity to express my sincere gratitude to CDO and CE administrative staff Laura Koller and Michael Lim for their support and help throughout my studies at MIT.

My heartfelt appreciation also goes to my dear friends: Chun Fan, Fitriani, Boon Hooi, and Yifeng. Without you guys, my stay in US would have been boring and uneventful. Thank you for lending me a helping hand when I lost my laptop just one week before the deadline for thesis submission. I would also like to extend my gratitude to other SMA and CDO classmate.

Lastly, I want to thank my parents and my brother who always been very supportive and understanding.
3.4.1 Projecting the partitions obtained at the coarsest level back to original graph $G_0$ .......................................................... 44
3.4.2 Refinement Step .......................................................... 45

4 Experimental Results for Graph Partition ....................................... 53
  4.1 Multilevel Graph Bisection Results ....................................... 54
    4.1.1 Coarsening Phase ...................................................... 54
    4.1.2 Uncoarsening Phase ................................................... 57
  4.2 Multilevel k-way Partition Results ...................................... 65
    4.2.1 Coarsening Phase ...................................................... 65
    4.2.2 Partitioning Phase .................................................... 66
    4.2.3 Uncoarsening Phase ................................................... 70

5 Experimental Results for Image Segmentation ................................ 73
  5.1 Weighting Function for Adjacency Matrix ............................... 74
  5.2 Coarsening Phase .......................................................... 75
    5.2.1 Using Ncut value as Measure of the Quality of Image Segmentation ..................... 82
  5.3 Partitioning Phase .......................................................... 83
  5.4 Uncoarsening Phase .......................................................... 87
    5.4.1 Inverse Powering Method ............................................. 89
    5.4.2 Greedy Local Refinement ............................................. 93
  5.5 Segmentation Results ....................................................... 96

6 Conclusion and Future Work .................................................. 105
  6.1 Conclusion ................................................................. 105
  6.2 Future Work ............................................................... 107

A Effects of Levels of Coarsening on the Ncut value and Segmentation Time for Graph Bisection .................................................... 109

B Graph Coarsening ............................................................... 111

C Refinement Method Comparisons ............................................. 117

D Effects of Levels of Coarsening on the Ncut value and Segmentation Time for k-Way Graph Partition ................................................. 123
E  Comparison between the Recursive Bisection and Multiclass Spectral Clustering Method  127

F  Effects of Levels of Coarsening on the Quality of Image Segmentation  131

G  Recursive Bisection Step  139

H  Natural Images Segmentation Data  141
List of Figures

2-1 Graph partitioning results for the normalized cut method. Link edges are weighted based on distance information. The method segments the graphs along the axis of symmetry. .................................................. 31

2-2 Minimum $Ncut = 0.0245$, and $threshold = 0.0074$ (shown in (c)). However, the graph partition using this criterion yields an asymmetrical graph partition. A symmetrical graph partition is obtained by $Ncut = 0.0258$ and $threshold = -0.0011$ (shown in (b)) .................................................. 32

2-3 Top Row: Graphs with two distinct groups. There are two discrete values in the Fiedler vector of each graph. Bottom Left: Graph with three distinct groups. There are three discrete values in the Fiedler vector of each graph. Bottom Right: Graph with four distinct groups. There are four discrete values in the Fiedler vector of each graph. .................................................. 32

2-4 Unweighted and weighted graph partitioning. An edge is built between the two groups with probability $p = 0.01$, 0.50 and 0.85. Unweighted graph partitioning fails for $p = 0.85$. .................................................. 33

2-5 First few smallest eigenvectors of a black image with a white region in the middle. Eigenvector #2 (Fiedler vector) has two discrete values, indicating that there are two distinct regions in the image, while the other three eigenvectors are almost continuous. .................................................. 34

2-6 First few smallest eigenvectors of an image with white, gray, and black regions. Eigenvector #2 (the Fiedler vector) has three discrete values, indicating that there are three distinct regions in the image. .................................................. 35

2-7 Segmentation of the image shown in Figure 2-5. Blue nodes and links represent the white region of the image, while red nodes and links represent the black region of the image .................................................. 35
2-8 Three levels recursive Ncut segmentation of a black image with a white region in the middle. The image is partitioned into 2, 4, and 8 groups for Level 1, Level 2, and Level 3, respectively.

3-1 Multilevel Clustering Framework consisted of three phases: 1) Coarsening, 2) Partition, and 3) Uncoarsening phase. The partition obtained from the coarse graph, \( G_{i+1} \) is refined and projected to the finer graph, \( G_i \). \[9, 2\]

3-2 A graph with unit edge weight. The edge weight of the link that connects the multinode with the neighboring node in the coarse graph is the sum of edge weights of the edges that it replaces.

3-3 Coarsening steps of a 16-node lattice structured graph. Two vertices of \( G_i \) form a multinode in the coarser graph, \( G_{i+1} \).

3-4 An uncoarsening step of a simple graph. The set of vertices grouped into a multinode will be assigned to the same partition as the multinode.

3-5 The greedy local refinement finds the move (shown in pink) along the boundary points (shown in green) that is associated with the largest decrease in the Ncut value. The point will then be swapped to the opposite group. These steps are iterated until there is no further improvement found for the partition.

3-6 Moving the node \( i \) from group 1 to group 2 results in cutting four edges (shown in blue) as compared to only two edges (shown in green). As a heuristic to the greedy refinement algorithm, only the nodes (among the boundary points) that has less or equal cut value is considered in finding the maximal gain in the Ncut value.

4-1 Effects of the number of coarsening levels on Ncut value. The quality of the partition obtained deteriorates as the number of coarsening level increases.

4-2 Effects of the number of coarsening levels on the execution time. The multilevel partitioning time is significantly shorter than partitioning \( G_0 \). The execution time decreases as the number of coarsening level increases. Note that the marginal improvement in execution time decreases with the amount of coarsening.

4-3 The process of coarsen the graph Crack from the original graph \( G_0 \) to the coarse graph \( G_5 \). (a) Original graph, \( G_0 \) (b)-(f) Coarse Graphs \( G_1, G_2, ..., G_5 \). As the graph is coarsened, the crack in the mesh ‘opens’ up.
4-4  The process of coarsen the graph *Tapir* from the original graph $G_0$ to the coarse graph $G_5$. (a) Original graph, $G_0$ (b)-(f) Coarse Graphs $G_1, G_2, ..., G_5$. The structure of the original problem (the head and body of the horse) is still preserved after 5 levels of coarsening.

4-5  Comparison between the Fiedler vector of $G_0$ with the converged vector obtained by the inverse powering method. All the vectors converged to their respective Fiedler vector with vector norm $< 1 \times 10^{-6}$.

4-6  Bisection's boundary points for graph a) *eppstein* b) *spiral*. Greedy local refinement is applied to these boundary points.

4-7  Segmentation of the graph *airfoil1* by a) normalized cut, b) projection, c) greedy local refinement, and d) combination of inverse powering method with greedy local refinement.

4-8  Segmentation of the graph *tapir* by a) normalized cut, b) projection, c) greedy local refinement, and d) combination of inverse powering method with greedy local refinement.

4-9  Effects of the number of coarsening levels on the Ncut value for partitioning the graph into four and eight segments ($k = 4$ and $k = 8$). In general, the quality of the partition deteriorates with increasing amount of coarsening.

4-10 Effects of the number of coarsening levels on execution time for partitioning the graph into four and eight segments ($k = 4$ and $k = 8$). In general, the running time required decreases with increasing amount of coarsening. The reduction in running time for the 8-way partition is less as compared to the 4-way partition.

4-11 Comparison between the recursive bisection and multiclass spectral clustering method for 2, 4, and 8 partitions for graph *airfoil1* and *tapir*. For the recursive bisection method, the method only has the choice of further splitting the subgraphs obtained at the previous recursion while the multiclass method can simultaneously segment the graph into the number of required partitions.

5-1  The top left corner shows an a synthetic image with a white square in the center against black background. The affinity plots for the four different weighting schemes are shown in column 2. The segmented results for the corresponding weighting schemes are shown in column 3. For this relatively simple image, all four weighting schemes yield the correct segmentation.
5-2 The top left corner shows an image of a bird. The affinity plots for the four different weighting schemes are shown in column 2. The segmented results for the corresponding weighting schemes are shown in column 3. The combined cue weighting scheme yields the best segmentation among the four schemes. For the exponential and Gaussian scheme, the image is segmented along the weak edges. However, these two schemes failed to segment the bird homogeneously as compared to the combined cue scheme. The linear weighting scheme yields the worst partition as the edge weight has slow a rate of decay. .......

5-3 The top left corner shows an image of the MIT dome. The affinity plots for the four different weighting schemes are shown in column 2. The segmented results for the corresponding weighting schemes are shown in column 3. The best partition is obtained by the combined cue weighting scheme. The two trees on the left and right and the field are correctly segmented. The partition quality given by the exponential and Gaussian schemes is similar, but worse than the combined cue scheme. The linear weighting scheme is not adequate in segmenting the image. ........................................ 78

5-4 Effects of the number of coarsening levels on the $Ncut$ value. The $Ncut$ value increases with the number of coarsening level. ........................................ 81

5-5 Effects of the number of coarsening levels on the execution time. The multilevel partitioning time is significantly shorter than segmenting the original image. The execution time decreases as the number of coarsening level increases. 81

5-6 Three observations can be made here: First, Fiedler method (multiclass spectral clustering) does not give the minimum $Ncut$ value; Second, even after two levels of coarsening, the lowest $Ncut$ value is 54.55% lower than than the Fiedler's $Ncut$ value; Third, the best partition to human eye is obtained not by Fiedler nor the lowest $Ncut$ value, but at $Ncut = 0.0006$. The time for segmenting the original image (of size 200x267) is 286.41 second. The segmentation time for two and three levels of coarsening are 78.15 and 36.47 second, respectively. In this example, the multilevel algorithm finds the best partition with 87.27% reduction in segmentation time. ...................... 84

14
5-7 For this image, the lowest $Ncut$ value is obtained by the Fiedler method (segmenting the original image). However, even though the $Ncut$ value at two levels of coarsening increased by 525%, the segmentation quality is the same as the one obtained by Fiedler. In both cases, the two flowers were segmented homogenously from its background (compare (b) and (c)). At ten levels of coarsening, the $Ncut$ value is increased by a massive 2225%. Only one flower is segmented homogenously while the other flower is group with one portion of the background. ................................. 85

5-8 Comparison of the segmentation results obtained by the recursive bisection and multiclass spectral clustering method for two images: MIT Dome and Heart. The recursive bisection method failed to segment the object in the image homogenously. The method keeps bisecting the graph into smaller parts, even though the subgraph contains the salient object. On the contrary, the multilevel spectral clustering method is able to segment the objects in the image homogenously. ................................. 88

5-9 Comparison of segmentation time for three schemes: partitioning the original image with multiclass spectral clustering; multilevel segmentation without refinement; and multilevel segmentation with inverse powering method. The multilevel segmentation without refinement (projection only) is significantly faster than the other two schemes. The multilevel level segmentation with inverse powering method is faster than the time required to segment the original image by multiclass spectral clustering. As the graph size increases, the time advantages of these two multilevel schemes are more evident. ................................. 91

5-10 Multilevel segmentation for the image baby with 10 levels of coarsening and the inverse powering as refinement method. The segmentation quality obtained by the inverse powering method (refer to (d)) is identical to partitioning the original image without multilevel (refer to (b)). Partitioning the original without multilevel requires 23.19 second while the multilevel with inverse powering only requires 4.14 second. ................................. 92
5-11 Multilevel segmentation for the image bride with 10 levels of coarsening and the inverse powering as refinement method. The segmentation quality obtained by the inverse powering method (refer to (d)) is identical to partitioning the original image without multilevel (refer to (b)). Partitioning the original without multilevel requires 38.97 second while the multilevel with inverse powering only requires 23.44 second.

5-12 The multilevel segmentation of the image panther without refinement (projection only) itself already yields an improvement of 28.45% in the $Ncut$ value over the Fiedler method. When the greedy local refinement algorithm is applied to the projected partition as the graph is being uncoarsened, the $Ncut$ value is further improved from 0.0171 to 0.0126, an improvement of 47.28% as compared to the $Ncut$ value given by the Fiedler method. However, even though the $Ncut$ value is reduced by a big margin, the two segmentations obtained with and without refinements are indeed similar (compare (c) and (d)).

5-13 The multilevel segmentation of the image Koala Bear without refinement (projection only) itself already yields an improvement of 20.81% in the $Ncut$ value over the Fiedler method. When the greedy local refinement algorithm is applied to the projected partition as the graph is being uncoarsened, the $Ncut$ value is further improved from 0.0449 to 0.0280, an improvement of 50.61% as compared to the $Ncut$ value given by the Fiedler method. However, even though the $Ncut$ value is reduced by a big margin, the two segmentations obtained with and without refinements are indeed similar (compare (c) and (d)).

5-14 First Column: Original Image, Second column: Fielder segmentation, Third column: Multilevel segmentation. The multilevel algorithm requires shorter segmentation time and offer similar, if not better, segmentation quality as compared to the Fielder method.

5-15 The multilevel algorithm requires shorter segmentation time and offer similar, if not better, segmentation quality as compared to the Fielder method.

B-1 The process of coarsen the graph $Airfoil1$ from the original graph $G_0$ to the coarse graph $G_5$. (a) Original graph, $G_0$ (b)-(f) Coarse Graphs $G_1, G_2, ..., G_5$

B-2 The process of coarsen the graph $Airfoil2$ from the original graph $G_0$ to the coarse graph $G_5$. (a) Original graph, $G_0$ (b)-(f) Coarse Graphs $G_1, G_2, ..., G_5$
B-3 The process of coarsen the graph Airfoil3 from the original graph $G_0$ to the coarse graph $G_5$. (a) Original graph, $G_0$ (b)-(f) Coarse Graphs $G_1, G_2, ..., G_5$

B-4 The process of coarsen the graph Parc from the original graph $G_0$ to the coarse graph $G_5$. (a) Original graph, $G_0$ (b)-(f) Coarse Graphs $G_1, G_2, ..., G_5$

B-5 The process of coarsen the graph Parcweb from the original graph $G_0$ to the coarse graph $G_5$. (a) Original graph, $G_0$ (b)-(f) Coarse Graphs $G_1, G_2, ..., G_5$

B-6 The process of coarsen the graph Spiral from the original graph $G_0$ to the coarse graph $G_5$. (a) Original graph, $G_0$ (b)-(f) Coarse Graphs $G_1, G_2, ..., G_5$

B-7 The process of coarsen the graph Eppstein from the original graph $G_0$ to the coarse graph $G_5$. (a) Original graph, $G_0$ (b)-(f) Coarse Graphs $G_1, G_2, ..., G_5$

B-8 The process of coarsen the graph Smallmesh from the original graph $G_0$ to the coarse graph $G_5$. (a) Original graph, $G_0$ (b)-(f) Coarse Graphs $G_1, G_2, ..., G_5$

C-1 Segmentation of the graph crack by a) normalized cut, b) projection, c) greedy local refinement, and d) combination of inverse powering method with greedy local refinement

C-2 Segmentation of the graph airfoil2 by a) normalized cut, b) projection, c) greedy local refinement, and d) combination of inverse powering method with greedy local refinement

C-3 Segmentation of the graph airfoil2 (Zoomed in)

C-4 Segmentation of the graph airfoil3 by a) normalized cut, b) projection, c) greedy local refinement, and d) combination of inverse powering method with greedy local refinement

C-5 Segmentation of the graph airfoil3 (Zoomed in)

C-6 The process of coarsen the graph parc. (a) Original graph, $G_0$ (b)-(f) Coarse Graphs $G_1, G_2, ..., G_5$

C-7 Segmentation of the graph parcweb by a) normalized cut, b) projection, c) greedy local refinement, and d) combination of inverse powering method with greedy local refinement

C-8 Segmentation of the graph spiral by a) normalized cut, b) projection, c) greedy local refinement, and d) combination of inverse powering method with greedy local refinement

C-9 Segmentation of the graph eppstein by a) normalized cut, b) projection, c) greedy local refinement, and d) combination of inverse powering method with greedy local refinement
C-10 Segmentation of the graph *smallmesh* by a) normalized cut, b) projection, c) greedy local refinement, and d) combination of inverse powering method with greedy local refinement ................................................. 122

E-1 Comparison between the recursive bisection and multiclass spectral clustering method for 2, 4, and 8 partitions for graph *crack*. ...................................................... 128

E-2 Comparison between the recursive bisection and multiclass spectral clustering method for 2, 4, and 8 partitions for graph *parc*. .................................................. 128

E-3 Comparison between the recursive bisection and multiclass spectral clustering method for 2, 4, and 8 partitions for graph *parcweb*. ............................. 129

E-4 Comparison between the recursive bisection and multiclass spectral clustering method for 2, 4, and 8 partitions for graph *spiral*. ........................................ 129

E-5 Comparison between the recursive bisection and multiclass spectral clustering method for 2, 4, and 8 partitions for graph *eppstein*. ........................................ 130

E-6 Comparison between the recursive bisection and multiclass spectral clustering method for 2, 4, and 8 partitions for graph *smallmesh*. ................................. 130

F-1 Segmentation results for multilevel segmentation for 1 to 10 levels of coarsening without refinement. First Column: *Landscape*, Second column: *Bird*, Third column: *Flower*. .................................................. 133

F-2 Segmentation results for multilevel segmentation for 1 to 10 levels of coarsening without refinement. First Column: *Skater*, Second column: *Grass*, Third column: *MIT Dome*. .................................................. 135

F-3 Segmentation results for multilevel segmentation for 1 to 10 levels of coarsening without refinement. First Column: *Panther*, Second column: *Butterfly*, Third column: *Taj Mahal*. .................................................. 137

G-1 Recursive Bisection ...................................................... 139

G-2 Recursive Bisection for Heart ........................................ 140
List of Tables

4.1 Graphs (from John Gilbert FTP site) used for evaluating the performance of multilevel graph partitioning algorithm. ........................................... 54
4.2 Size of graph, $|V_i|$ for 5 levels of coarsening ............................... 55
4.3 Results of the inverse powering method. All projected vector converged to the Fiedler vector of original graph $G_0$. The base partitioning vector obtained from the partitioning phase is close to the Fiedler vector as only a few power iterations are required for the vector to converged. This method offer huge reduction in running time while giving the same partition quality as solving a generalized eigenvalue problem on $G_0$. ........................................... 59
4.4 Results of the greedy local refinement method. This refinement scheme does not guarantee an improvement in the partition quality. Nevertheless, the execution time is less than solving a generalized eigenvalue problem on $G_0$ (except for the graph crack). ........................................... 62
4.5 Results of the combined method of inverse powering method with greedy local refinement. The running time of this combined method is faster and produces a partition of equal or better quality than solving a generalized eigenvalue problem on $G_0$. ........................................... 64
4.6 Comparison between the $Ncut$ value and segmentation time obtained by recursive bisection and multiclass spectral clustering for $k = 8$. The multiclass spectral clustering method outperforms the recursive bisection method in terms of speed and quality. ........................................... 69
4.7 Results of inverse powering method for partitioning the test graphs into four partitions. All project vectors converged to the Fiedler vectors of $G_0$. The method offers time reduction for all the graphs except crack and smallmesh. The partitioning quality obtained by the inverse powering method is the same as partitioning using Fiedler method (without multilevel implementation). 70
4.8 Results of the combination of the inverse powering method with greedy local refinement for partitioning the test graphs into four partitions. All graphs except spiral have improvement in the $Ncut$ value, ranging from 1.37 to 8.57%. The $Ncut$ value for spiral is the same as the Fiedler method. This could be that the partition is already optimal. However, the segmentation time of the combination of inverse powering with greedy refinement is longer as compared to the time required for the spectral partition of the original graph.

5.1 Size of the graph, $|V_i|$ for 5 levels of coarsening.

5.2 $Ncut$ Values for various levels of coarsening (no refinement).

5.3 Execution time for various levels of coarsening (no refinement).

5.4 Comparison between the $Ncut$ value and segmentation time obtained by recursive bisection and multiclass spectral clustering for $k = 8$. The test images are resized to about 100x100 in size.

5.5 Summary of the segmentation times and $Ncut$ values for segmenting the image bird by three schemes: partitioning the original image with multiclass spectral clustering (Fiedler); multilevel segmentation without refinement (Project); and multilevel segmentation with inverse powering method (Power). Three levels of coarsening are applied to both of the multilevel segmentation schemes.

A.1 $Ncut$ Values for various levels of coarsening (no refinement).

A.2 Execution time for various levels of coarsening (no refinement).

D.1 $Ncut$ Values for various levels of coarsening ($k = 4$).

D.2 $Ncut$ Values for various levels of coarsening ($k = 8$).

D.3 Execution time for various levels of coarsening ($k = 4$).

D.4 Execution time for various levels of coarsening ($k = 8$).

H.1 Segmentation data for natural images segmentation.
Chapter 1

Introduction

1.1 Background

1.1.1 Graph Partition

Graph partition is the process of segmenting a graph into multiple regions. In general, graph partition can be loosely categorized into two groups: spectral and non-spectral. For the non-spectral graph partition, there are max-flow-min-cut, isoperimetric, and $k$-means method while spectral graph partition includes the normalized cut, spectral rounding, and multiclass spectral clustering.

These days, graph partition has applications in: [17]

- Neural networks
- Data mining, document indexing
- Bioinformatics, systems biology
- Information theory
- Predictive control, robotics
- Parallel computing

For non-spectral graph partition, one way to segment a graph is by using the maximum-flow-minimum-cut method. Imagine that the nodes are the depots of FedEx logistics and the weight of the links represents the maximum traffic flow allowed in their respective links. Links that have large differences in node values will carry a small weight, thus having a
limited flow as compared to one where the difference in node values is small. The optimal partition is obtained by finding the set of nodes such that the outgoing flow from these nodes is the minimum among all the possible combinations of nodes. This way the graph is now partitioned into two segments. The process is repeated until the numbers of desired partitions are obtained.

Grady and Schwartz [5, 6] proposed the isoperimetric graph partitioning method. In this method, the graph is viewed as an electric circuit. First, a node in the graph is grounded. Subsequently, the graph is partitioned into clusters by solving a linear system. The method is faster and more stable than the spectral graph partitioning.

For $k$-means clustering [17], the first step is to find the centroids of the $k$ initial set either by random generation or any heuristic. Next, the nodes in the graph are clustered into its closest centroid. A new set of $k$ centroids are computed for the new partition. The process continues until there is no further exchanges of nodes between clusters.

For the spectral graph partitioning, despite being a subject that has been known for nearly two decades, interest in this topic only regains its momentum in 2000 when Shi and Malik [16] proposed the idea of normalized cut. Their contributions were significant because their method was able to address the major problems faced in the maximum-flow-minimum-cut method. Minimum cut favors cutting small sets of isolated nodes in the graph while the normalized cut does not.

Tolliver and Miller [18] further improved the normalized cut concept by introducing the spectral rounding method. The link weights are reweighted until the graph disconnects into the prescribed number of partitions. The spectral rounding method compares favorably to normalized cut and yields better segmentation.

Yu and Shi [19] proposed the multiclass spectral clustering method to discretize the first $k$ eigenvectors of the graph Laplacian matrix into $k$ discrete partitions. The method finds the discrete near-global optimal partition. Yu and Shi also reported that their method is robust to random initialization.

1.1.2 Image Segmentation

Image segmentation is the partition of an image into multiple regions based on various criteria such as color, intensity, and texture. One of the most important applications of image segmentation is in medical imaging. Doctors often require their patients to undergo various types of screening to obtain images that help them analyze the medical status of the patients (such as Magnetic Resonance Imaging, MRI).
For example, locating a tumor by visually inspecting tons of images is a daunting task. Through image segmentation, the software can assist the doctors by highlighting potential areas. Furthermore, using image segmentation, various components of the image can be differentiated, allowing the procedures such as measuring the volume of the tissue. These procedures can help surgeons anticipate surgical operations in advance. Other areas where image segmentation is applicable are face recognition and machine vision.

An image is formed by millions of pixels containing information such as the intensity and Red-Green-Blue (RGB) values. This information is transformed into a graph, analogous to the 2-D lattice structure. Each node on the graph represents a pixel in the corresponding image. The links between the nodes of the lattice are weighted such that each weight carries data regarding similarity between two nodes.

However, results from image segmentation are not always completely reliable. There are still many computational issues that remain unsolved. Some of the typical questions are what is the criterion for a good partition and how can such a partition be computed efficiently. Given the same image, two different people may partition the image differently.

Lately, many different approaches to image segmentation have arisen. Additional segmentation cues such as the texture, contour, and object boundaries [4, 12, 14] are used to improve the quality of image segmentation. Apart from low-level segmentation (such as brightness, color and motion), mid- or high-level knowledge about symmetries of objects or object models is also incorporated. This prior knowledge of the image improves the reliability of the segmentation. For example, this information about tumor size and shape is programmed into the computer, and it helps the program to recognize the tumor in the image more accurately, thus improving the reliability of the segmented image.

The quest to find an efficient algorithm to segment an image with the best possible accuracy is an ongoing effort. Riding on the tide of great advancement in computational power, the potential for growth in image segmentation is huge. Many different areas of image segmentation are yet to be explored.

1.2 Motivation

While the spectral graph partitioning method gives high quality segmentation, segmenting large graphs by the spectral method is computationally expensive. Hendrickson and Leland [7] proposed a multilevel scheme that significantly reduces the partitioning time of large graphs.
However, the multilevel algorithms proposed in [7, 9, 8] refine the partition by an iterative greedy local refinement scheme. Dhillon et. al [2] pointed out that local refinement schemes have the tendency of trapping the partition in poor local minima.

When such a situation occurs, the partition obtained by the multilevel algorithm will have a lower quality than partitioning the original graph. Hence, there is a need to find a multilevel algorithm that offers both speed and quality that is as good as or better than partitioning the large graph by the spectral method.

1.3 Objective

The main objectives of this research are as follows:

1. Develop a multilevel graph partitioning algorithm that incorporates the inverse powering method with greedy local refinement, which offers better partition quality than partitioning the original graph (without multilevel).

2. Study the efficiency of the multilevel algorithm proposed to graph bisection and k-way graph partitioning in terms of speed and partition quality.

3. Investigate how does the multilevel image segmentation compares to spectral image segmentation without the multilevel implementation (Fiedler method).

1.4 Thesis Organization

The thesis is organized in the following way. Chapter 2 is a review of the partitioning power of the spectral graph partitioning method. In this chapter, I demonstrate the capabilities and characteristics of the method by applying it to simple test graphs. In addition, I also present the analysis of the eigenvectors used in segmenting simple black and white images.

Chapter 3 introduces the general framework for the multilevel spectral graph partitioning algorithm. Here, I present a scheme to construct the adjacency matrix, $W$ and degree matrix, $D$ for the coarse graphs. The scheme takes advantage of the special structure of the Coarsening Matrix, $CM$, which stores the matching information for coarsening a graph from $G_i$ to $G_{i+1}$, to construct the affinity matrices of the coarse graphs. In addition, I also introduce a refinement method that combines the inverse powering method with a greedy local refinement scheme.
In Chapter 4, I show the experimental results for graph partition obtained by using the multilevel graph partitioning algorithm presented in Chapter 3. First, I analyze the results for multilevel graph bisection, which partitions a graph into two segments. Next, I study the results for multilevel $k$-way partitioning, which partitions a graph into $k$ segments. The results of the breakdown of the three phases - coarsening, partitioning, and uncoarsening - are investigated in detail.

Chapter 5 discusses the experimental results for image segmentation. The chapter begins with the analysis of four edge weighting scheme for building the image affinity matrix. Subsequently, I looked into the suitability of using the $Ncut$ value as a measure for the quality of image segmentation. The advantages and disadvantages of using the recursive bisection and multiclass spectral clustering method in the partitioning phase are discussed. In addition, the results for the inverse powering and greedy refinement method are presented. The chapter ends with a presentation of the segmentation results of natural images.

The thesis concludes with a summary of this work and recommendations for future work.
Chapter 2

Partitioning Power of Spectral Graph Method: A Review

2.1 The Normalized Cut

Shi and Malik [16] improve the minimum cut criteria, \( \text{cut}(A, B) = \sum_{u \in A, v \in B} w(u, v) \), by normalizing the cut value with the total connection from the nodes in one partition to all other nodes in the graph. The normalized cut formulation is as follows: [16]

\[
\text{Ncut}(A, B) = \frac{\text{cut}(A, B)}{\text{assoc}(A, V)} + \frac{\text{cut}(A, B)}{\text{assoc}(B, V)},
\]

where \( \text{assoc}(A, V) = \sum_{u \in A, t \in V} w(u, t) \).

They have proven that the normalized cut formulations, after relaxing the constraint \( x(i) \in \{1, -b\}, b \in \mathbb{R} \), can be viewed as a generalized eigenvalue problem of \( (D-W)\bar{x} = \lambda D\bar{x} \). The Fiedler vector, \( \bar{x} \), contains the segmentation information. Ideally, the Fiedler vector should contain two discrete values, and give a trivial discrete segmentation. However, segmentation of natural images by the normalized cut method may yield a continuous eigenvector \( \bar{x} \), which makes the image segmentation nontrivial.
2.2 Edge-Weighting Scheme

2.2.1 Weighting Scheme for Graph Partition

A graph can be weighted or unweighted. For a weighted graph, the edge weights play an important role in determining how the graph is to be partitioned. Equation 2.2 shows the edge-weighting scheme for graph partitioning proposed by Shi and Malik [16]. Note that under this scheme, the edge weights decay exponentially with nodal distance. Since the probability of having two nodes that are far away but in the same group is small, the edge weight is set to zero when the distance between the two nodes exceeds $r$.

$$W(i, j) = \begin{cases} 
\exp \left( -\frac{\|X(i) - X(j)\|^2}{\sigma^2} \right) & \text{if } \|X(i) - X(j)\|_2 \leq r \\
0 & \text{otherwise}
\end{cases} \quad (2.2)$$

2.2.2 Weighting Scheme for Image Segmentation

For spectral image segmentation, an image is transformed into a graph with each vertex representing a pixel in the image. Edges built between the vertices are weighted to reflect the similarity between the pixels. For example, if two pixels belong to a same object in the image, the edge between these two pixels should be given a high weightage. It is unlikely for the spectral method to cut along strongly linked edges. Cutting along edges with high weightage will increase the normalized cut (Ncut) value.

In low level image segmentation, there is no prior knowledge about the image. Therefore, local grouping cues such as the pixel intensity, distance between pixels and contour information of an image are used to weight the edges of a graph. Pixels belonging to an object usually have low variation in the color intensity. In addition, pixels that are far apart have lower probability of belonging to the same object. Hence, the edge weight should be inversely proportional to the difference in the intensity and distance between pixels.

The adjacency matrix, $W$ (also known as the similarity matrix) stores the edge weights between pixels. $W(i, j)$ contains the edge weight for the link between vertex $i$ and $j$. Shi and Malik [16] proposed three weighting schemes that use intensity and distance information as grouping cues: Linear, Exponential, and Gaussian weighting schemes. In addition to the intensity and distance information, Timothée et. al. [1] proposed a new weighting scheme that also includes the intervening contour information. These four weighting schemes for building the adjacency matrix are as follows:
1. Linear Weighting Scheme: [16]

\[ W(i, j) = \left(1 - \frac{|I(i)-I(j)|}{\sigma_I \times 255}\right) \left(1 - \frac{|X(i)-X(j)|}{\sigma_X \times r}\right) \] (2.3)

2. Exponential Weighting Scheme: [16]

\[ W(i, j) = \exp \left(- \frac{|I(i)-I(j)|}{\sigma_I \times 255}\right) \exp \left(- \frac{|X(i)-X(j)|}{\sigma_X \times r}\right) \] (2.4)

3. Gaussian Weighting Scheme: [16]

\[ W(i, j) = \exp \left(- \frac{|I(i)-I(j)|^2}{\sigma_I \times 255^2}\right) \exp \left(- \frac{|X(i)-X(j)|^2}{\sigma_X \times r^2}\right) \] (2.5)

4. Combined Cue of Intensity, Distance and Contour: [1]

\[ W_L(i, j) = \exp \left(- \frac{|I(i)-I(j)|^2}{\sigma_I \times 255^2}\right) \exp \left(- \frac{|X(i)-X(j)|^2}{\sigma_X \times r^2}\right) \] (2.6)

\[ W_C(i, j) = \exp \left(-\max_{x \in \text{line}(i, j)} \left| \frac{E(x)}{\sigma_C} \right|^2 \right) \] (2.7)

\[ W_C(i, j) = \sqrt{W_L(i, j) \times W_C(i, j)} + \alpha W_C(i, j) \] (2.8)

where \( I(i) \) is the pixel intensity of node \( i \)
\( X(i) \) is the position of node \( i \)
\( r \) is the maximum distance between two pixels for which link are built
\( \sigma_I, \sigma_X, \) and \( \sigma_C \) are the parameters for intensity, distance and contour, respectively
\( \text{line}(i, j) \) is a straight line joining node \( i \) and \( j \)
\( E(x) \) is the edge strength at location \( x \).

Among the linear, exponential and gaussian weighting schemes, edge weights for the gaussian scheme decay at the fastest rate. On the other hand, edge weights for the exponential scheme drop at a faster rate as compared to the linear weighting scheme. The higher the decay rate, the lower will be the cut value across pixels with large difference in pixel intensity.

For images with small variation in pixel intensity (such as an animal camouflaged in an environment), the inclusion of the contour information into the grouping cue helps to
improve the segmentation. If the straight line joining node $i$ and $j$ crosses the contour line, it is less likely for these two pixels to be in the same group. The edge strength between these two nodes is high, and thus, the link between them is weak.

The results obtained for graph partitioning and image segmentation using these weighting schemes with the normalized cut method are presented in Section 2.3 and 2.4 respectively.

### 2.3 Graph Partitioning

The normalized cut method provides a convenient way to partition a network flow graph without solving a traditional maximum-flow-minimum-cut problem. Minimum cuts tend to cut isolated nodes in a graph while the normalized cut avoid this possibility by normalizing the cut criterion. This is because partitioning isolated nodes in a graph will yield high $Ncut$ value. Hence, the normalized cut method strikes a balance between finding an optimal segmentation and avoiding segmenting the isolated nodes.

Graph partitioning results for the normalized cut method are shown in Figure 2-1. The edges are weighted based on the distance information. Links are removed for two nodes that have distances that exceeds $r = 1$ for this example. It is observed that for symmetrical graphs, generally the cut that yields the lowest $Ncut$ value is obtained by segmenting the graph along the axis of symmetry.

However, a case where the normalized cut method obtains the minimum $Ncut$ value through an asymmetrical partition is shown in Figure 2-2. Although the minimum $Ncut$ value is 0.0245, symmetrical partition leads to a higher $Ncut$ value at 0.0258 and threshold $= -0.0011$. Therefore, finding the minimum $Ncut$ value does not guarantee the best partition. First, there is numerical error and roundoff when computing the minimum $Ncut$ value iteratively (finding the global optimum $Ncut$ is an NP-Complete problem). Secondly, the constraint that requires the solution $\vec{x}$ to have only two discrete values is not always satisfied.

If there exist distinct groups that only have links among themselves, the Fiedler vector has few discrete values depending on the number of isolated groups. Figure 2-3 shows examples of graphs with distinct groups, and their segmentation. The grouping information is reflected in their Fiedler vector. The normalized cut segmentation will segment out one of these isolated groups. Note that the minimum $Ncut$ for a graph with distinct groups should be zero since there is no link between the groups. However, the minimum $Ncut$ obtained experimentally is on the order of $1e-45$ due to numerical roundoff. The remaining groups can be partitioned recursively until all the groups are found based on the 'jumps' detected
in the Fiedler vector. For example, a Fiedler vector with two distinct ‘jumps’ implies that there are three distinct groups in the graph.

Figure 2-4 shows the comparison between unweighted and weighted graph partitioning. A link is built between the isolated groups with probability $p$. The unweighted graph partitioning performed as expected for $p = 0.01$ and 0.50, which is to segment the two circles. However, as more and more links are built between the two groups (such as $p = 0.85$), the unweighted graph partitioning is unable to find the two groups. The weighted graph partitioning works for all values of $p$ (regardless of the number of links between two groups). The Fiedler vector for the weighted graph partitioning has two discrete values while the Fiedler vector for the unweighted graph partitioning becomes continuous for higher $p$ values.

![Graph partitioning results for the normalized cut method.](image)

Figure 2-1: Graph partitioning results for the normalized cut method. Link edges are weighted based on distance information. The method segments the graphs along the axis of symmetry.
Figure 2-2: Minimum $Ncut = 0.0245$, and $threshold = 0.0074$ (shown in (c)). However, the graph partition using this criterion yields an asymmetrical graph partition. A symmetrical graph partition is obtained by $Ncut = 0.0258$ and $threshold = -0.0011$ (shown in (b))

Figure 2-3: Top Row: Graphs with two distinct groups. There are two discrete values in the Fiedler vector of each graph. Bottom Left: Graph with three distinct groups. There are three discrete values in the Fiedler vector of each graph. Bottom Right: Graph with four distinct groups. There are four discrete values in the Fiedler vector of each graph.
Figure 2-4: Unweighted and weighted graph partitioning. An edge is built between the two groups with probability $p = 0.01$, $0.50$ and $0.85$. Unweighted graph partitioning fails for $p = 0.85$.

2.4 Normalized Cut applied to Image Segmentations

The normalized cut method can also be applied to image segmentation. In addition to the distance information between two nodes, the information on pixel intensity difference is also used in the edge weighting. Since the probability of a group having similar pixel intensity is high, the edges that have small variation in pixel intensity difference are weighted more heavily.
2.4.1 Two-Way \textit{Ncut}

The segmentation information is contained in the first few smallest eigenvectors of the generalized eigenvalue problem \((D - W)\vec{x} = \lambda D\vec{x}\). The first eigenvector is a constant vector with eigenvalue zero and has no segmentation information.

Figure 2-5 and Figure 2-6 show the first few Fiedler vectors of two images: 1) a black image with a white region in the middle, and 2) an image with white, gray, and black regions. The second eigenvector partitions the image into two groups by assigning two discrete values. However, as seen in Figure 2-6, the eigenvector #2 can have more than two discrete values. In this example, there are three discrete values since the image contains three distinct regions (white, gray, and black regions). The higher eigenvectors can be used to partition the image into more groups [16]. For instance, eigenvector #3 can be used to partition an image into three groups.

Figure 2-7 shows the segmentation of a black image with a white region in the middle using the Fiedler vector, \(\vec{x}\). Indeed, the Fiedler vector is successful in identifying the two distinct groups (the white and black regions) in the image. The link weights along the edges of the white region decay exponentially fast due to the sharp difference in pixel intensities. In other words, the black and white regions in the image are separated since the link weights along the edges are almost zero. This property has given rise to two discrete values of the Fiedler vector \(\vec{x}\).

Figure 2-5: First few smallest eigenvectors of a black image with a white region in the middle. Eigenvector #2 (Fiedler vector) has two discrete values, indicating that there are two distinct regions in the image, while the other three eigenvectors are almost continuous.
Figure 2-6: First few smallest eigenvectors of an image with white, gray, and black regions. Eigenvector #2 (the Fiedler vector) has three discrete values, indicating that there are three distinct regions in the image.

Figure 2-7: Segmentation of the image shown in Figure 2-5. Blue nodes and links represent the white region of the image, while red nodes and links represent the black region of the image.

2.4.2 Recursive Two-Way Ncut

The two-way Ncut method can be extended to segment an image recursively until the prescribed number of partitions is achieved. Figure 2-8 shows the result for a three level recursive Ncut of a black image with a white center (Figure 2-5). At Level 1, the black and white regions are segmented (shown in red and blue). Subsequently, the black and white groups are subpartitioned into two segments, respectively. At Level 3, there will be a total of eight segmented groups.
Figure 2-8: Three levels recursive $Ncut$ segmentation of a black image with a white region in the middle. The image is partitioned into 2, 4, and 8 groups for Level 1, Level 2, and Level 3, respectively.
Chapter 3

Multilevel Spectral Clustering: The Algorithm

3.1 Multilevel Clustering Framework

The multilevel graph partitioning method is generalized into three phases: Coarsening Phase, Partitioning Phase, and Uncoarsening Phase [7, 9, 11]. Figure 3-1 shows a typical implementation of the multilevel clustering method. Given a graph $G_0$, the graph is coarsened successively from $G_0$ to $G_1$, $G_2$, until $G_m$ by collapsing vertices and edges. The coarsening phase terminates when the size of the coarsest graph is only a prescribed fraction of the size of the original graph $G_0$.

At the coarsest level, the graph $G_m$ is partitioned into $k$ segments using any graph partitioning techniques. These partitioning techniques are 1) multiclass spectral clustering [19]; 2) Spectral Rounding [18]; 3) bisection method [2]; 4) region growing algorithm [9], and 5) Kernighan-Lin algorithm [10, 3, 7]. In addition, Karypis and Kumar [8] found that a recursive bisection algorithm such as the normalized cut method [16] produces good initial partitions.

The partition obtained at the coarsest level is then projected to the finer graph successively from the graph $G_{m-1}$, $G_{m-2}$, back to $G_0$. Although the optimal partition of the coarse graph may be far away from the optimal partition for the fine graph, applying a refinement algorithm periodically largely resolves this problem [7]. The refinement step is essential to improve the partition's quality, as a finer graph has more degrees of freedom in clustering the vertices [9]. A general multilevel partitioning method overview is shown in Algorithm 1 (page 39).
In this thesis, I present a scheme to construct the adjacency matrices, W and degree matrices for the coarse graphs. In addition, I introduce a refinement algorithm which combines the inverse power method with a greedy local refinement scheme.

Figure 3-1: Multilevel Clustering Framework consisted of three phases: 1) Coarsening, 2) Partition, and 3) Uncoarsening phase. The partition obtained from the coarse graph, $G_{i+1}$ is refined and projected to the finer graph, $G_i$ ∀i. [9, 2]

3.2 Phase 1: Coarsening

3.2.1 Coarsening Schemes

In the coarsening phase, the graph $G_0$ is coarsened by collapsing vertices and edges such that $|V_0| > |V_1| > |V_2| > ... > |V_m|$, where $|V_i|$ is the size of graph $G_i$. The graphs must be coarsened in such a way the connectivity information of the original graph is preserved [9, 8]. This is to ensure that the nature of the original problem is not changed. A set of vertices in the fine graph, $G_i$ is combined to form a multinode of the coarse graph, $G_{i+1}$.
Algorithm 1 Multilevel Partitioning Method Overview [7, 11]

Coarsening Phase:
The graph $G_0$ is transformed into a sequence of smaller graphs $G_1, G_2, ..., G_m$ such that $|V_0| > |V_1| > |V_2| > ... > |V_m|$, where $|V_i|$ is the size of graph $G_i$.

Partitioning Phase:
The coarsest graph $G_m = (V_m, E_m)$ is partitioned into 2 or more segments by any graph partitioning methods.

Uncoarsening Phase:
The partition $P_m$ of $G_m$ is projected back to $G_0$ by going through a sequence of intermediate partitions $P_{m-1}, P_{m-2}, ..., P_1, P_0$.

The edges within the multinode are removed while those edges that incident out of the multinode is preserved in $G_{i+1}$. In the case where two or more vertices of the multinode have links to the same neighboring node, the edge weight of the link that connects the multinode with that neighboring node in $G_{i+1}$ is the sum of the weights of the edges it replaces. Hendrickson and Leland [7] illustrate this case by collapsing one edge of a triangular graph with unit edge weight (Figure 3-2).

![Figure 3-2: A graph with unit edge weight. The edge weight of the link that connects the multinode with the neighboring node in the coarse graph is the sum of edge weights of the edges that it replaces.](Fine Graph Coarse Graph)

Coarse graphs, $G_{i+1}$ can be constructed from fine graphs, $G_i$ by computing a matching of $G_i$. A matching is defined as a set of edges where no two edges are incident to the same vertex [11, 7]. Vertices of $G_i$ form multinodes of $G_{i+1}$ according to the pairing obtained from the matching. The unmatched vertices are carried over to $G_{i+1}$. Since the objective of the coarsening phase is to shrink the size of the graph, finding the maximal matching is desirable. A matching is maximal if adding a new edge destroys the matching [9, 8]. The coarsening step terminates when the size of the coarse graph, $G_m$ is only a fraction of the size of original graph, $|V_0|$ or when the ratio of the size of $G_i$ to $G_{i+1}$, $|V_i|/|V_{i+1}|$ is close to
Figure 3-3 shows the coarsening steps of a 16-node lattice structured graph.

Figure 3-3: Coarsening steps of a 16-node lattice structured graph. Two vertices of $G_i$ form a multinode in the coarser graph, $G_{i+1}$.

Various graph coarsening algorithms have been proposed heretofore. These include random matching, heavy edge matching, light edge matching, and heavy clique matching. A brief summary of these matching algorithms is as follows [9]:

1. **Random Matching**
   All the vertices in the graph are visited once in a random order. The visited vertex is paired with an adjacent vertex that has not been paired. These two vertices will then be flagged. If the visited vertex has been flagged, the vertex will be ignored and the process continues. This algorithm requires an effort of $O(|E|)$, where $|E|$ is the number of edges.

2. **Heavy Edge Matching**
   Heavy edge matching is similar to the random matching. However, instead of randomly pairing the visited vertex with a random adjacent vertex, heavy edge matching finds the adjacent unmatched vertex with the highest edge weight among all unmatched incident edges. Computing heavy edge matching also requires an effort of $O(|E|)$.

3. **Light Edge Matching**
   Light edge matching is in similar spirit with the heavy edge matching except that light edge matching finds the adjacent unmatched vertex with the lowest edge weight among all unmatched incident edges. Computing light edge matching also requires an effort of $O(|E|)$. 
4. Heavy Clique Matching

Heavy clique matching is yet another variant of random matching. An unmatche
vertex is matched with an adjacent vertex with the largest edge density. The moti
behind this matching scheme is that subgraphs that are clique or almost clique are
unlikely to be cut. A clique is a fully connected subgraph of an unweighted graph. The
dge density between two vertices \( u \) and \( v \) is given by:

\[
\text{Edge cut} = \frac{2(ce(u) + ce(v) + ew(u,v))}{(vw(u) + vw(v)) \times (vw(u) + vw(v) - 1)}
\]  

(3.1)

where \( vw(u) \) = weight of the vertex
\( ce(u) \) = sum of the weight of the collapsed edges of \( u \)
\( ew(e) \) = weight of the edge

3.2.2 Constructing the Adjacency matrix, W and Degree matrix, D for the coarse graphs

Here, I present an algorithm for constructing the adjacency matrices, \( W \) and degree matrices, \( D \) for the coarse graphs \( G_1, G_2, \ldots \) to \( G_m \). The information of the matching obtained from any of the matching schemes described in the previous section is transferred to a matrix named Coarsening Matrix, \( CM \). For the convenience of storing the matching information at different levels, \( CM \) is initialized as a cell in Matlab.

\( CM\{i\} \) stores the coarsening information between \( G_{i-1} \) and \( G_i \). The size of \( CM\{i\} \) is \( |G_{i-1}| \times |G_i| \). The columns of \( CM\{i\} \) represent the vertices of the coarse graph \( G_i \) while the rows of \( CM\{i\} \) represent the vertices of the fine graph \( G_{i-1} \). If, for example, node 4 and 8 of \( G_{i-1} \) are grouped into node 3 of \( G_i \), the entries of \( CM\{i\}(4,3) \) and \( CM\{i\}(8,3) \) will be set to 1. The coarsening matrix for the transformation between the graph shown in Figure 3-3(b) and (c) is as follows:

41
$CM\{2\} =$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Due to the special structure of the coarsening matrix, the adjacency matrix, $W\{i+1\}$ can be obtained by a matrix multiplication of $W\{i+1\} = CM\{i\}^T \times W\{i\} \times CM\{i\}$. The heavy edge matching algorithm for constructing the coarsening matrix cell is shown in Algorithm 2 (page 43) while the algorithm for building adjacency matrices and degree matrices is shown in Algorithm 3 (page 43).

### 3.3 Phase 2: Partitioning

At the coarsest level, the graph $G_m = (V_m, E_m)$ is partitioned into $k$ segments using any of the graph partitioning techniques. This partitioning phase is fast because the coarsest graph, $G_m$ is substantially smaller in size as compared to the original graph, $G_0$. In this thesis, I focus on the spectral clustering method for the base partition. The two spectral clustering techniques tested are the recursive bisection (normalized cut), and multiclass spectral clustering.

The recursive normalized cut method first bisects a graph into two partitions. These two partitions are then further bisected recursively until the prescribed number of partitions are obtained. One of the main drawbacks of the recursive bisection method is that the normalized cut value is only minimized for the two particular partitions during a bisection step. In other words, the overall normalized cut value for the $k$-way partition is not optimized.

For the multiclass spectral clustering, the first step is to solve for a generalized eigenvalue problem on the coarsest graph, $G_m$ to obtain its first $k$ smallest eigenvectors. Subsequently, the $k$ eigenvectors are discretized into $k$ discrete partitions by the rounding method. In this work, I use the multiclass spectral clustering code published online by Yu and Shi\(^1\). I made use of the discretization function in the code to obtain $k$ discrete partitions.

---

\(^1\)available online at http://www.cis.upenn.edu/ jshi/software/
**Algorithm 2** Heavy Edge Matching

**Input:** Adjacency matrix of original graph $W^{1}$

**Output:** Coarsening Matrix cell, $CM$

1: $i \leftarrow 1$
2: while true do
3:     $k \leftarrow 1$
4:     initialize vector, flag
5:     initialize Coarsening Matrix cell, $CM$
6:     $W_{temp} \leftarrow \text{upper triangular}(W^{i})$
7:     for $j = 1$ to $\text{length}(W^{i})$ do
8:         if $\text{flag}(j) == 1$ then
9:             continue
10:     end if
11:     [$\text{max.value}, \text{max.index}$] $\leftarrow \text{max}(W_{temp}(j,:))$
12:     if $\text{max.value}! = 0$ then
13:         $CM^{i}(\text{max.index}, k) \leftarrow 1$
14:         $\text{flag}(q) \leftarrow 1$
15:         $W_{temp}(:, q) \leftarrow 0$
16:     end if
17:     $CM^{i}(j, k) \leftarrow 1$
18:     $k \leftarrow k + 1$
19: end for
20: if $\text{length}(W^{i + 1}) < 0.05 \times \text{length}(W^{1})$ or $i >= \text{max.level}$ then
21:     break
22: end if
23: end while

**Algorithm 3** Build Adjacency matrix, $W$, and Degree matrix, $D$, for coarse graphs

**Input:** Adjacency matrix of original graph $W^{1}$ and Coarsening matrices, $CM$

**Output:** Adjacency and Degree matrices of coarse graphs, $W^{\text{level} + 1}, D^{\text{level} + 1}$

1: initialize cell $CM$
2: for $k = 1$ to $\text{level}$ do
3:     $W^{k + 1} \leftarrow CM^{k}T \times W^{k} \times CM^{k}$
4:     for $i = 1$ to $N$ do
5:         $D^{k + 1}(i, i) \leftarrow \sum_{j=1}^{N} W_{ij}$
6:     end for
7: end for
3.4 Phase 3: Uncoarsening

3.4.1 Projecting the partitions obtained at the coarsest level back to original graph $G_0$

In the uncoarsening phase, the partitions obtained from the partitioning phase are projected to the finer graph successively from the graph $G_{m-1}, G_{m-2}$, back to $G_0$. The set of collapsed vertices which form a multinode will be assigned the same grouping in the finer graph. For example, if node 2 and 6 in graph $G_i$ are grouped to form multinode 3 in $G_{i+1}$, node 2 and 6 will be in the same partition as multinode 3.

The $k$ partitioning vectors of the coarsest graph $G_m, z_0, z_1, z_2, \ldots, z_k$ are projected through the levels by multiplying these partitioning vectors with the coarsening matrix:

$$Z_{G_i} = CM\{\text{level} - (i - 1)\} \times Z_{G_{i+1}}$$

(3.2)

where $Z_{G_{i+1}}$ is the matrix of partitioning vectors of graph $G_{i+1}$.

$$Z_{G_{i+1}} = \begin{bmatrix} \vdots & \vdots & \vdots \\ z_1 & z_2 & \cdots & z_k \\ \vdots & \vdots & \vdots \end{bmatrix}$$

(3.3)

Again, this matrix-vector multiplication is possible due to the special structure of the coarsening matrix. The matrix-vector multiplication of the coarsening matrix with the partitioning vectors can be viewed as a linear combination of the columns of the coarsening matrix. Since a row of the coarsening matrix sums to one, the linear combination gives exactly the same grouping as the multinode. Shown in Figure 3-4 is a simple case of a bipartition ($k = 2$) to illustrate the process of projecting the partitioning vector from $G_{i+1}$ to $G_i$.

![Figure 3-4: An uncoarsening step of a simple graph. The set of vertices grouped into a multinode will be assigned to the same partition as the multinode.](image)
In this example, nodes 1 and 2 are grouped into multinode 1, while nodes 3 and 4 are grouped into multinode 2. During the uncoarsening phase, nodes 1 and 2 will be assigned the same group as multinode 1 (shown in red) whereas nodes 3 and 4 will be assigned the same group as multinode 2 (shown in blue). The linear combination of the columns of the coarsening matrix with the partitioning vector is:

\[
\begin{bmatrix}
1 & 0 \\
1 & 0 \\
0 & 1 \\
0 & 1 \\
\end{bmatrix} \times \begin{bmatrix}
1 \\
0 \\
0 \\
\end{bmatrix} = \begin{bmatrix}
1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 \\
\end{bmatrix} \times 1 + \begin{bmatrix}
0 \\
1 \\
0 \\
\end{bmatrix} \times 0 = \begin{bmatrix}
1 \\
0 \\
0 \\
\end{bmatrix}
\]

Since the fine graphs have more degrees of freedom in clustering the nodes, the projected partitions from \( G_m \) to the finer graphs \( G_{m-1}, G_{m-2}, ..., G_0 \) are not the optimal partitions for those finer graphs. If the partitions are projected without refinement, the partitions will be further and further away from the optimal at every uncoarsening step. Therefore, it is necessary to refine the graph’s partitions as we uncoarsen the graph.

### 3.4.2 Refinement Step

In a conventional multilevel graph clustering scheme \([7, 9, 8, 2, 11]\), refinement steps are performed on the projected partitions as the graph is being uncoarsened. The quality of a partition is enhanced by swapping the vertices from one cluster to another, in search for a move that lowers the edge cut or normalized cut value.

However, the local refinement algorithm does not guarantee to improve the partitions such that the quality is better than solving a generalized eigenvalue problem on the original graph \( G_0 \) (without multilevel). The local refinement algorithm finds the move that has the largest decrease in the normalized cut value. Nevertheless, moving a point that increases the normalized cut value may then lead to a situation where the next move decreases the normalized cut value even more than moving the point associated with the maximum gain \([7]\).

Hendrickson and Leland \([7]\) also pointed out that the local refinement scheme must be fast and effective for the multilevel graph partitioning technique to be useful. Here, I present a refinement scheme that combines: 1) inverse powering method, and 2) greedy local refinement algorithm. The former utilizes the projected eigenvectors of the coarsest graph, \( G_m \) to obtain the Fiedler vectors of the original graph, \( G_0 \) while the latter improves the partition’s quality by swapping the grouping of boundary points to lower the normalized cut value. Combining
these two refinement techniques yields an algorithm that is faster than solving the generalized eigenvalue problem on the original graph, $G_0$ and at the same time produces a better quality partition.

**Inverse Powering Method**

At the coarsest level, the generalized eigenvalue system of is solved to obtain the first $k$ smallest eigenvectors of the coarsest graph $G_m$. Subsequently, these vectors are projected to the finer graphs $G_{m-1}$, $G_{m-2}$, back to $G_0$ by injection. The shorter vector is projected into a longer vector (for the finer graph) by inserting the vector value into the corresponding position in the higher level vector.

For example, if node 5 in the coarse graph is a supernode of node 8 and 11 in the finer graph, the eigenvector value in position 5 is copied to position 8 and 11 of the longer vector. The vector is then normalized after every projection. The algorithm for the inverse powering method is shown in Algorithm 4 on page 47.

At the finest level, the projected vector and the Rayleigh quotient value are used as a starting eigenvector and eigenvalue for the inverse powering method. For graph bisection ($k = 2$), the projected vector from the coarsest graph is a good starting point for the inverse powering method. The vector is close the Fiedler vector of the original graph because the graph is coarsened in such a way that the connectivity information of the original graph is preserved. Hence, the power method converges fast to the Fiedler vectors of the graph $G_0$. Since the projected vector converged to the Fiedler vector of $G_0$, the partition quality obtained by the inverse powering method is identical to the one obtained by partitioning the original graph $G_0$.

However, for the $k$-way partition, the starting eigenvector for the inverse iteration may converge to the higher or lower eigenvector of $G_0$. If such a situation occurs, the number of partitions obtained will be $k$ less the number of independent eigenvectors. Instead of using the Rayleigh quotient as the starting eigenvalue, the eigenvalue of the previous eigenvector is used. If the vector converges to the lower eigenvectors, the inverse iteration is repeated with a higher starting eigenvalue between the previous eigenvalue and the Rayleigh quotient of the starting vector. In this way, we can ensure that the first $k$ smallest eigenvectors of $G_0$ are obtained.
Algorithm 4 Inverse Powering Method for Graph Bisection

Input: Projected partition, \( z \)
Output: Converged Fiedler vector of fine graph, \( x \)

1: \( b \leftarrow z \) %assign projected partition, \( z \) to \( b \)
2: \( \mu \leftarrow \beta \times \frac{z^T(D-W)z}{z^TDz} \)
3: \( d \leftarrow \) diagonal of \( D \)
4: \( K \leftarrow D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}} \)
5: while true do
6: \( b \leftarrow (K-\mu I)^{-1}b \)
7: \( b \leftarrow \) normalize \( b \)
8: \( \mu \leftarrow b^T Kb \)
9: \( b_{\text{old}} \leftarrow b \)
10: \( b \leftarrow (K-\mu I)^{-1}b \)
11: \( b \leftarrow \) normalize \( b \)
12: %Iterate until eigenvector converged
13: if \( \text{norm}(|b_{\text{old}}|-|b|) < \epsilon \) then
14: \( \text{break} \)
15: end if
16: end while
17: \( x \leftarrow D^{-\frac{1}{2}}b \)
18: return \( x \)

Greedy Local Refinement Algorithm

In the greedy local refinement scheme, points are swapped from one group to another to improve the quality of the partition. Due to the nature of the refinement scheme, these points usually reside on the boundary of a cut. Hence, to avoid excessive computation, the algorithm first computes the nodes (boundary points) that have edges cut by the segmentation.

First, the boundary points are moved one by one to the opposite group to find the maximum decrease in the normalized cut value. Second, the point associated with the maximum decrease will be swapped to the opposite group (refer to Figure 3-5). I shall refer to these two processes as one refinement iteration. The refinement iteration terminates when there is no further improvement in the partition’s quality. Moving an additional point will instead increase the normalized cut value.

The algorithm for computing the boundary points is shown in Algorithm 5. The partitioning vector, \( z \), which is a 0-1 vector, is first discretized to a -1 and 1 vector, \( x \). The index of the boundary points is found by looking for negative entries in the matrix \( \text{Edge} \) obtained
by the matrix multiplication of:

\[ \text{Edge} = d \times W \times d \] (3.4)

where \( d \), the diagonal matrix of the vector \( x \) is:

\[ d = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \] (3.5)

The rows of the Edge matrix represent the nodes in a graph. If there is any negative entry along a row in the matrix, the node corresponding to that row is a boundary point.

\[ \text{Edge}(i, j) = x_i \times W(i, j) \times x_j = \begin{cases} w_{ij} & \text{if both } i \text{ and } j \text{ is in the same group} \\ -w_{ij} & \text{if } i \text{ and } j \text{ are in different groups} \\ 0 & \text{if there is no link between } i \text{ and } j \end{cases} \]

The cut for a boundary point \( i \) is defined as follows: \( D_{1i} = \sum W_{ij} \) for \( j \) connected to \( i \) and in group 1; \( D_{2i} = \sum W_{ij} \) for \( j \) connected to \( i \) and in group 2. As a heuristic to the greedy refinement algorithm, only the nodes among the boundary points that have \( D_{1i} \leq D_{2i} \) are considered in the refinement step. The algorithm for the greedy refinement algorithm is shown in Algorithm 6 (page 51).

The greedy algorithm terminates when there is no further improvement in the normalized value. For graphs that have many iteration steps, the time advantage may vanish. By setting the maximum number of iterations for the greedy algorithm, the time for refinement step can be controlled at the expense of less improvement in the normalized cut value.

The above greedy refinement can be extended to the \( k \)-way graph partition by repeating the algorithm for each partition. For example, when refining a 4-way partition, we can refine the boundary for each partition with the other three partitions (viewing the other partitions as as one partition) until there is no further improvement.

**Combination of Inverse Powering method with Greedy Local Refinement**

In the combined method, first, the vector obtained at the coarsest level is projected to the original graph \( G_0 \). Power iteration is performed on the vector until it converges to
Figure 3-5: The greedy local refinement finds the move (shown in pink) along the boundary points (shown in green) that is associated with the largest decrease in the $Ncut$ value. The point will then be swapped to the opposite group. These steps are iterated until there is no further improvement found for the partition.

Figure 3-6: Moving the node $i$ from group 1 to group 2 results in cutting four edges (shown in blue) as compared to only two edges (shown in green). As a heuristic to the greedy refinement algorithm, only the nodes (among the boundary points) that has less or equal cut value is considered in finding the maximal gain in the $Ncut$ value.
Algorithm 5 Compute Boundary Points

**Input:** Adjacency matrix, W and partition vector, z

**Output:** Indices of boundary points, index, and Edge cut vector, D1

1: \( x \leftarrow \text{discretize } z \text{ to a -1 and 1 vector} \)
2: \( d \leftarrow \text{Diagonal matrix of } x \)
3: \( \text{Edge} \leftarrow d \times W \times d \)
4: \( \text{index} \leftarrow \text{column indices of negative entries of the matrix } \text{Edge} \)
5: \( \text{Edge} \leftarrow \text{Edge(index,:)} \)
6: \( \text{for } i = 1 \text{ to } N \text{ do} \)
7: \( D1_i \leftarrow \sum_{j=1}^{N} (\text{Edge} > 0)_{ij} \)
8: \( \text{end for} \)
9: \( \text{return } D1 \text{ and index} \)

the Fiedler vector of \( G_0 \). Subsequently, the greedy refinement algorithm is applied to the partition obtained from the inverse powering method.

Since refinement is performed on the projected partitioned obtained by the Inverse Powering method, the partition quality will be equal or better than the partition obtained by the spectral partition of \( G_0 \).
Algorithm 6 Greedy Local Refinement for Graph Bisection

Input: Adjacency matrix, $W$; Degree matrix, $D$; partitioning vector, $z$; and maximum refinement iteration, $maxIteration$

Output: refined partitioning vector, $z$

1: $x \leftarrow$ discretize to a -1 and 1 vector
2: $d \leftarrow$ Diagonal of matrix $D$
3: $[index, D_1] \leftarrow \text{Compute\_Boundary\_Points}$
4: $m \leftarrow 1$
5: while true do
6:  $D_2 \leftarrow d(index) - D_1$
7:  for $i = 1$ to $N$ do
8:      $temp_i \leftarrow d(index)_i \times x(index)_i$
9:      $\Delta k_i \leftarrow -\frac{temp_i}{\sum d}$
10:     end for
11:     $temp \leftarrow \sum \frac{d}{\sum d}$
12:    for $i = 1$ to $N$ do
13:       $k_i \leftarrow temp + \Delta k_i$
14:       $b_i \leftarrow \frac{k_i}{1-k_i}$
15:    end for
16:    for $j = 1$ to length($index$) do
17:       $X(:,j) = x$
18:       $X(index(j),j) \leftarrow -X(index(j),j)$
19:    end for
20:    for $i = 1$ to length($index$) do
21:       $y_i \leftarrow (1 + x_i) - b_i(1 - x_i)$
22:       $\Delta Ncut_i \leftarrow current\_Ncut - \frac{y_i^T(D-W)y_i}{y_i^T D y_i}$
23:    end for
24:   $[\max\_\Delta Ncut, ind\_max] \leftarrow \max(\Delta Ncut)$
25:  if $\max\_\Delta Ncut <= 0$ || $m >= max\_Iteration$ then
26:     refinement\_Iteration $\leftarrow m$; break;
27:  end if
28: $m \leftarrow m + 1$
29: %Move the point associated with maximum decrease in $Ncut$
30:  $x(index(ind\_max)) \leftarrow -x(index(ind\_max))$
31:  $[i,j] \leftarrow \text{find}(W(index,:), > 0)$
32:  $\text{index\_boundary} \leftarrow \text{unique}(j)$ %function unique finds the unique elements of a vector
33:  $W\_boundary \leftarrow W(index\_boundary, index\_boundary)$
34:  $x\_boundary \leftarrow x(index\_boundary)$
35:  $[ind, D_1] \leftarrow \text{Compute\_Edge}(W\_boundary, x\_boundary)$
36:  $index \leftarrow index\_boundary(ind)$
37:  $current\_Ncut \leftarrow \text{Compute\_Ncut}(W, D, x)$
38: end while
39: $z \leftarrow (x > 0)$
Chapter 4

Experimental Results for Graph Partition

In this chapter, I present the results obtained by the multilevel spectral clustering algorithm described in Chapter 3. Graphs from various finite element problems are used as test graphs for the multilevel graph clustering algorithm. The characteristics of these test graphs\(^1\) are summarized in Table 4.1. One of the most important applications for multilevel graph partitioning is in parallel computing. The finite element meshes are partitioned into segments such that the load to each core is optimal with the least amount of data communication [7].

All the experiments are performed on an Apple MacBook with a 2.16Ghz Intel Core 2 duo processor and 2Gigabytes of memory, running on Windows Vista Enterprise edition via boot camp. All times recorded are in seconds. The normalized cut (\(Ncut\)) criterion is used as a gauge to the partition quality of a graph. In addition, as the multilevel algorithm is randomized in nature, I choose to implement the matching scheme in a fixed order (from first vertex to the last vertex) instead of matching the vertices randomly.

This chapter is divided into two sections: multilevel graph bisection, and multilevel \(k\)-way partition. First, I discuss the performance and findings on the multilevel graph bisection. In the multilevel graph bisection, a graph is segmented into two partitions by solving a generalized eigenvalue problem on the coarsest graph, \(G_m\). Second, I present the results of the multilevel \(k\)-way partition. In the \(k\)-way partition, a graph is segmented into \(k\) partitions by recursive bisection and multiclass spectral clustering.

\(^1\)Available at FTP site of John Gilbert and the Xerox Corporation at ftp://ftp.parc.xerox.com/pub/gilbert/meshes.tar.Z
Table 4.1: Graphs (from John Gilbert FTP site) used for evaluating the performance of multilevel graph partitioning algorithm.

<table>
<thead>
<tr>
<th>Graph</th>
<th>Nodes</th>
<th>Edges</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>airfoil1</td>
<td>4253</td>
<td>28831</td>
<td>Three-element airfoil by Barth and Jespersen</td>
</tr>
<tr>
<td>airfoil2</td>
<td>4720</td>
<td>32164</td>
<td>Large fragment of Barth/Jespersen 3-element airfoil</td>
</tr>
<tr>
<td>airfoil3</td>
<td>15606</td>
<td>107362</td>
<td>Four-element airfoil by Barth and Jespersen</td>
</tr>
<tr>
<td>crack</td>
<td>5120</td>
<td>35326</td>
<td>Crack propagation mesh from CMU</td>
</tr>
<tr>
<td>parc</td>
<td>1240</td>
<td>7950</td>
<td>Bounded aspect ratio mesh from Xerox PARC</td>
</tr>
<tr>
<td>parcweb</td>
<td>1939</td>
<td>12731</td>
<td>Non-obtuse mesh from Xerox PARC</td>
</tr>
<tr>
<td>spiral</td>
<td>1200</td>
<td>7582</td>
<td>Spiral mesh by Gilbert and Simon</td>
</tr>
<tr>
<td>tapir</td>
<td>1024</td>
<td>6716</td>
<td>Non-obtuse mesh by Bern</td>
</tr>
<tr>
<td>eppstein</td>
<td>547</td>
<td>3679</td>
<td>Bounded aspect ratio mesh by Eppstein</td>
</tr>
<tr>
<td>smallmesh</td>
<td>136</td>
<td>844</td>
<td>Small fragment of Barth/Jespersen 3-element airfoil</td>
</tr>
</tbody>
</table>

4.1 Multilevel Graph Bisection Results

4.1.1 Coarsening Phase

In the coarsening phase, a graph is coarsened until the number of nodes of the coarse graph is down to a prescribed fraction of the total number of nodes in the original graph. To do a parametric analysis of the coarsening of a graph with respect to the partition quality and execution time, a graph is coarsened from 1 to 10 levels. All the graphs are coarsened by Heavy Edge matching and no refinement is performed on the projected partition during the uncoarsening phase.

Table A.1 shows the normalized cut value for various coarsening levels while Table A.2 shows the execution time needed. These results are presented graphically in Figure 4-1 and 4-2, respectively. The execution time includes the time needed for all the three phases: coarsening, partitioning, and uncoarsening. Data is not available for the graph smallmesh at Level 9 and 10 as the graph is coarsened down to one node in the coarsest graph.

From Figure 4-1, we see that the quality of a partition deteriorates (higher Ncut value) as the graph goes through more coarsening levels. For example, the Ncut value for the graph eppstein is increased from an increment of 22.60% for five levels of coarsening to a massive increment of 2358.97% for ten levels of coarsening. This is not surprising as the more coarsening is done on a graph, the further away the coarse graph will be from the original problem. Too much coarsening will change the nature of the problem.

From Figure 4-2, we observed that the multilevel partitioning execution time is signif-
icantly shorter than the time required for solving a generalized eigenvalue problem on the original graph, $G_0$. The execution time decreases with increasing number of coarsening levels. For example, the reduction in execution time for one level of coarsening for the graph *spiral* is 30.80% while the execution time is reduced by 95.89% for ten levels of coarsening. Note that the marginal improvement in execution time decreases with the amount of coarsening. This means that after a graph has been coarsened for many levels, there is no further improvement in the execution time by applying more and more coarsening to the graph.

On one hand, it is desirable to apply more coarsening levels to a graph as the execution time is much shorter than partitioning graph $G_0$. On the other hand, the more the coarsening, the worse will be the partition. Hence, in order to strike a balance between the partition quality and execution time, I chose to limit the coarsening to a maximum of five levels. In all the experiments, the graph $G_0$ is coarsened to 5% of the total graph nodes $|V_0|$ or restricted to five levels of coarsening.

The size of graphs, $|V_i|$ for $i = 1, 2, ..., 5$ is shown in Table 4.2. After five levels of coarsening, the coarse graph, $G_5$ has only a small fraction (about 3–5%) of the number of nodes of original graph $|V_0|$. The number of nodes for the coarse graphs of *crack*, *parcweb*, and *tapir* are still about 20% of the original graph as the maximum number of coarsening level is five. The ratio of the number of nodes in the fine graph to the coarse graph, $\frac{|V_i|}{|V_{i+1}|}$ is about 2 during the first few levels of coarsening. This means that the number of nodes of the graph is almost halved at each level of coarsening. However, this ratio drops to 1 after coarsening the graph for many levels. This is because as the graph gets coarser, the number of available matching decreases.

### Table 4.2: Size of graph, $|V_i|$ for 5 levels of coarsening

| Graph    | $|V_0|$ | Level | $|V_i|$ | Level | $|V_i|$ | Level | $|V_i|$ | Level | $|V_i|$ | Level | $|V_i|$ | Level | $|V_i|$ | Level | $|V_i|$ |
|----------|--------|-------|--------|-------|--------|-------|--------|-------|--------|-------|--------|-------|--------|-------|--------|-------|
| airfoil1 | 4253   | 1 | 2127   | 2 | 1067   | 3 | 535    | 4 | 271    | 5 | 141    | 6 | 3.32   |       |        |
| airfoil2 | 4720   | 1 | 2360   | 2 | 1183   | 3 | 597    | 4 | 302    | 5 | 156    | 6 | 3.31   |       |        |
| airfoil3 | 15606  | 1 | 7806   | 2 | 3910   | 3 | 1962   | 4 | 990    | 5 | 504    | 6 | 3.23   |       |        |
| crack    | 5120   | 1 | 3343   | 2 | 2280   | 3 | 1668   | 4 | 1312   | 5 | 1110   | 6 | 21.68  |       |        |
| parc     | 1240   | 1 | 644    | 2 | 347    | 3 | 197    | 4 | 118    | 5 | 75     | 6 | 6.05   |       |        |
| parcweb  | 1939   | 1 | 1221   | 2 | 811    | 3 | 578    | 4 | 446    | 5 | 367    | 6 | 18.93  |       |        |
| spiral   | 1200   | 1 | 623    | 2 | 333    | 3 | 183    | 4 | 109    | 5 | 71     | 6 | 5.92   |       |        |
| tapir    | 1024   | 1 | 650    | 2 | 438    | 3 | 316    | 4 | 245    | 5 | 204    | 6 | 19.92  |       |        |
| eppstein | 547    | 1 | 277    | 2 | 143    | 3 | 75     | 4 | 41     | 5 | 24     | 6 | 4.39   |       |        |
| smallmesh| 136    | 1 | 69     | 2 | 36     | 3 | 19     | 4 | 11     | 5 | 7      | 6 | 5.15   |       |        |
Effects of the number of coarsening levels on the partition's quality

Figure 4-1: Effects of the number of coarsening levels on $N_{cut}$ value. The quality of the partition obtained deteriorates as the number of coarsening level increases.

Effects of coarsening levels on execution time

Figure 4-2: Effects of the number of coarsening levels on the execution time. The multilevel partitioning time is significantly shorter than partitioning $G_0$. The execution time decreases as the number of coarsening level increases. Note that the marginal improvement in execution time decreases with the amount of coarsening.
Shown in Figure 4-3 and 4-4 is the evolution of coarsening a graph from $G_0$ to $G_5$ for the graph *crack* and *tapir*, respectively. From these figures, we see that the coarse graphs still preserve the structure of the original problem. The connectivity information of the original graph $G_0$ is carried to the coarsest graph $G_m$. From Figure 4-3, we see that the crack in the mesh ‘opens’ up as the graph is coarsened. In addition, from Figure 4-4, the head and body of the horse in the mesh *tapir* is still preserved even after 5 levels of coarsening. Refer to Appendix B for a full list graph coarsening examples.

![Figure 4-3: The process of coarsen the graph Crack from the original graph $G_0$ to the coarse graph $G_5$. (a) Original graph, $G_0$ (b)-(f) Coarse Graphs $G_1, G_2, ..., G_5$. As the graph is coarsened, the crack in the mesh ‘opens’ up.](image)

### 4.1.2 Uncoarsening Phase

In the uncoarsening phase, the partition obtained from the coarsest graph $G_m$ is projected back to the original graph $G_0$. As the projected partition is not the optimal partition for the fine graph, refinement steps are performed to improve the quality of the partition. Here, I compare the performance of three refinement algorithms: inverse powering method, greedy local refinement, and the combination of these two methods.

In order to have a fair comparison of these three refinement methods, the same coarsening algorithm and base partitioning method is used for all experiments. Heavy Edge matching is used in the coarsening phase while the base partition is obtained by solving a generalized eigenvalue problem on the coarsest graph $G_m$. 

57
Figure 4-4: The process of coarsen the graph Tapir from the original graph $G_0$ to the coarse graph $G_5$. (a) Original graph, $G_0$ (b)-(f) Coarse Graphs $G_1, G_2, ..., G_5$. The structure of the original problem (the head and body of the horse) is still preserved after 5 levels of coarsening.

**Inverse Powering Method**

The base partitioning vector, obtained by solving a generalized eigenvalue problem on the coarsest graph $G_m$, is projected to $G_0$ through the injection method as described in Chapter 3. Power iteration is performed to converge the projected vector to the Fiedler vector of graph $G_0$.

Table 4.3 shows the results obtained for multilevel clustering with inverse powering. As seen from this table, the inverse powering method offers a reduction of up to 89% (for the graph spiral) in running time as compared to the time needed to partition the original graph $G_0$. However, the running time for smallmesh, a 136-node graph, only improved by 18.83%. In other words, applying a multilevel graph partitioning method to large graphs is more advantageous than running the algorithm on small graphs.

In addition, all the projected base partitioning vectors converged to the Fiedler vector of their original graph with a vector norm of less than $1 \times 10^{-6}$. These base partitioning vectors are in fact very close to the Fiedler vector as only a few power iterations are needed to converge these vectors. The fact that the vector converges in a few iterations shows that the coarse graph still preserves the essential connectivity information of the original graph. Figure 4-5 shows the comparison between Fiedler vector and the converged vector obtained from the inverse powering method.
Table 4.3: Results of the inverse powering method. All projected vector converged to the Fiedler vector of original graph $G_0$. The base partitioning vector obtained from the partitioning phase is close to the Fiedler vector as only a few power iterations are required for the vector to converged. This method offers huge reduction in running time while giving the same partition quality as solving a generalized eigenvalue problem on $G_0$.

<table>
<thead>
<tr>
<th>Graph</th>
<th>$Ncut$</th>
<th>$%$ Diff</th>
<th>Time (second)</th>
<th>$%$ Diff</th>
<th>Norm</th>
<th>Power Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fiedler</td>
<td>multilevel</td>
<td></td>
<td>Fiedler</td>
<td>multilevel</td>
<td></td>
</tr>
<tr>
<td>airfoil1</td>
<td>0.0094</td>
<td>0.0094</td>
<td>0.0000</td>
<td>3.8227</td>
<td>1.1422</td>
<td>-70.1213</td>
</tr>
<tr>
<td>airfoil2</td>
<td>0.0135</td>
<td>0.0135</td>
<td>0.0000</td>
<td>3.7131</td>
<td>1.3768</td>
<td>-62.9192</td>
</tr>
<tr>
<td>airfoil3</td>
<td>0.0066</td>
<td>0.0066</td>
<td>0.0000</td>
<td>34.1678</td>
<td>9.8856</td>
<td>-71.0675</td>
</tr>
<tr>
<td>crack</td>
<td>0.0193</td>
<td>0.0193</td>
<td>0.0000</td>
<td>4.4354</td>
<td>2.2939</td>
<td>-48.2807</td>
</tr>
<tr>
<td>parc</td>
<td>0.0125</td>
<td>0.0125</td>
<td>0.0000</td>
<td>1.4849</td>
<td>0.2990</td>
<td>-79.8630</td>
</tr>
<tr>
<td>parcweb</td>
<td>0.0096</td>
<td>0.0096</td>
<td>0.0000</td>
<td>1.7957</td>
<td>0.5950</td>
<td>-66.8649</td>
</tr>
<tr>
<td>spiral</td>
<td>0.0044</td>
<td>0.0044</td>
<td>0.0000</td>
<td>2.5705</td>
<td>0.2746</td>
<td>-89.3185</td>
</tr>
<tr>
<td>tapir</td>
<td>0.0080</td>
<td>0.0080</td>
<td>0.0000</td>
<td>0.7115</td>
<td>0.2642</td>
<td>-62.8606</td>
</tr>
<tr>
<td>eppstein</td>
<td>0.0407</td>
<td>0.0407</td>
<td>0.0000</td>
<td>0.2876</td>
<td>0.1070</td>
<td>-62.8091</td>
</tr>
<tr>
<td>smallmesh</td>
<td>0.0679</td>
<td>0.0679</td>
<td>0.0000</td>
<td>0.0707</td>
<td>0.0574</td>
<td>-18.8347</td>
</tr>
</tbody>
</table>
Figure 4-5: Comparison between the Fiedler vector of $G_0$ with the converged vector obtained by the inverse powering method. All the vectors converged to their respective Fiedler vector with vector norm $< 1 \times 10^{-6}$. 
Since the base partitioning vectors converged to the Fiedler vector, the partition qualities obtained are the same for the inverse powering method and normalized cut method (solving for the Fiedler vector on \( G_0 \)). Hence multilevel clustering with the inverse powering method is fast and yet achieve the same partition quality as partitioning the original graph \( G_0 \) without multilevel.

**Greedy Local Refinement**

The greedy local refinement improves a partition’s quality by swapping nodes from one group to another. At each iteration, the greedy local refinement algorithm finds the move (among the boundary points) which is associated with the maximum decrease in the \( N\text{cut} \) value. The point will then be moved to the opposite group and the algorithm continues until there is no further improvement in the partition’s quality.

Figure 4-6 shows the boundary points along the bisection for *eppstein* and *spiral*. Among all the nodes in a graph, the most probable move which improves a partition lies in these boundary points. Hence, in order to have a fast and efficient refinement algorithm, the greedy refinement is performed on these boundary points.

Table 4.4 shows the results obtained for multilevel clustering with greedy local refinement. From the results, we see that the greedy local refinement does not guarantee an improvement in the partition’s quality. For the graph *tapir*, the \( N\text{cut} \) value is increased from 0.0080 to 0.0128, an huge increment of 61.04%. Only four (airfoil1, airfoil2, crack, and smallmesh) out of the ten graphs has improved \( N\text{cut} \) value.

The greedy local refinement does not guarantee an improvement in the partition’s quality. Moving the point which increases the \( N\text{cut} \) value may then lead to a higher reduction in the \( N\text{cut} \) value as opposed to moving the point associated with the local maximum gain. This method finds a local minimum, and the partition may be driven further and further away from the global optimal partition.

The running time for the graph *crack* is 7.29% slower than partitioning the original graph \( G_0 \). The execution time is long because there are many moves that offer a small reduction (often on the order of \( 10^{-6} \)) in the \( N\text{cut} \) value. Hence, there will be many refinement iterations before there is no further improvement in the partition. In this case, the number of refinement iterations should be restricted in order to ensure that the greedy refinement algorithm is fast.
Table 4.4: Results of the greedy local refinement method. This refinement scheme does not guarantee an improvement in the partition quality. Nevertheless, the execution time is less than solving a generalized eigenvalue problem on $G_0$ (except for the graph crack).

<table>
<thead>
<tr>
<th>Graph</th>
<th>Ncut</th>
<th>Time (second)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fiedler</td>
<td>multilevel</td>
</tr>
<tr>
<td>airfoil1</td>
<td>0.0094</td>
<td>0.0090</td>
</tr>
<tr>
<td>airfoil2</td>
<td>0.0135</td>
<td>0.0128</td>
</tr>
<tr>
<td>airfoil3</td>
<td>0.0066</td>
<td>0.0067</td>
</tr>
<tr>
<td>crack</td>
<td>0.0193</td>
<td>0.0191</td>
</tr>
<tr>
<td>parc</td>
<td>0.0125</td>
<td>0.0152</td>
</tr>
<tr>
<td>parcweb</td>
<td>0.0096</td>
<td>0.0104</td>
</tr>
<tr>
<td>spiral</td>
<td>0.0044</td>
<td>0.0052</td>
</tr>
<tr>
<td>tapir</td>
<td>0.0080</td>
<td>0.0128</td>
</tr>
<tr>
<td>eppstein</td>
<td>0.0407</td>
<td>0.0432</td>
</tr>
<tr>
<td>smallmesh</td>
<td>0.0679</td>
<td>0.0645</td>
</tr>
</tbody>
</table>

Figure 4-6: Bisection’s boundary points for graph a) eppstein b) spiral. Greedy local refinement is applied to these boundary points.
Combination of Inverse Powering method with Greedy Local Refinement

In this combined method, first, the base partitioning vector leads to the Fiedler vector of $G_0$. Subsequently, the greedy local refinement is used to improve the partition’s quality. As such, the partition’s quality will be equal or better than than solving a generalized eigenvalue problem on $G_0$.

Table 4.5 shows the results obtained for multilevel clustering with the combined method of inverse powering method with greedy local refinement. The $Ncut$ values for four graphs (parcweb, tapir, eppstein, and smallmesh) do not decreases even thought the greedy refinement algorithm is applied. This could be either the partition is already at optimum or the heuristic in the greedy refinement algorithm fails to find the point that potentially decrease the $Ncut$ value.

For spiral, the quality of the partition given by this combined method improved by 14.27%. On the other hand, the partition quality given by the greedy local refinement worsens by 19.01%. Both methods still offer a reduction in running time by 88.01% and 92.52%, respectively. In this case, the greedy local refinement has driven the partition further and further away from the optimal partition. On the other hand, the improvement in $Ncut$ value for airfoil1 is more for the greedy refinement method as compared to the combined method. In this case, the greedy refinement algorithm finds a local minima with a lower $Ncut$ value.

For smallmesh, the $Ncut$ value improved by 5.08% for the greedy refinement method while there is no improvement in the $Ncut$ value for the combined method. For the combined method, the greedy refinement is performed on the partition obtained by the inverse powering method. The fact that the greedy refinement does not offer any improvement means that the partition is trapped in a local minima.

In short, the greedy refinement algorithm may have the ability to improve a partition more than the combined method, but the improvement is not guaranteed. The combined method of inverse powering and greedy local refinement is better in the sense that the method is fast and guarantees a partition quality which is better or equal to the quality given by solving a generalized eigenvalue problem on $G_0$.

Figure 4-7 and 4-8 shows the partition obtained from 1) normalized cut, 2) projection, 3) greedy local refinement, and 4) the combination method for the graph airfoil1 and tapir. Since the partition obtained by the normalized cut method is the same as the partition obtained by the inverse powering method, the inverse powering method’s partition is not shown in these figures. Refer to Appendix C for a full list of the graph partitioning results.
Table 4.5: Results of the combined method of inverse powering method with greedy local refinement. The running time of this combined method is faster and produces a partition of equal or better quality than solving a generalized eigenvalue problem on $G_0$.

<table>
<thead>
<tr>
<th>Graph</th>
<th>$Ncut$</th>
<th>Time (second)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fiedler</td>
<td>multilevel</td>
</tr>
<tr>
<td>airfoil1</td>
<td>0.0094</td>
<td>0.0092</td>
</tr>
<tr>
<td>airfoil2</td>
<td>0.0135</td>
<td>0.0133</td>
</tr>
<tr>
<td>airfoil3</td>
<td>0.0066</td>
<td>0.0062</td>
</tr>
<tr>
<td>crack</td>
<td>0.0193</td>
<td>0.0187</td>
</tr>
<tr>
<td>parc</td>
<td>0.0125</td>
<td>0.0119</td>
</tr>
<tr>
<td>parcweb</td>
<td>0.0096</td>
<td>0.0096</td>
</tr>
<tr>
<td>spiral</td>
<td>0.0044</td>
<td>0.0038</td>
</tr>
<tr>
<td>tapir</td>
<td>0.0080</td>
<td>0.0080</td>
</tr>
<tr>
<td>eppstein</td>
<td>0.0407</td>
<td>0.0407</td>
</tr>
<tr>
<td>smallmesh</td>
<td>0.0679</td>
<td>0.0679</td>
</tr>
</tbody>
</table>

Figure 4-7: Segmentation of the graph airfoil1 by a) normalized cut, b) projection, c) greedy local refinement, and d) combination of inverse powering method with greedy local refinement.
4.2 Multilevel k-way Partition Results

In the multilevel k-way graph partition, a graph is segmented into k partitions. The multilevel framework for the k-way graph partition is similar to that of the multilevel graph bisection. At the base clustering, the coarsest graph is partitioned into k partitions either by the recursive bisection or multilevel spectral clustering method. Hereafter, I shall refer to the procedure of partitioning the original graph without using the multilevel implementation as the ‘Fiedler’ method.

4.2.1 Coarsening Phase

Regardless of the number of segments required, the coarsening phase for the multilevel graph bisection and k-way partition is the same. The coarsening steps for the graphs are shown in Appendix B. The test graphs are coarsened from 1 to 10 levels with the intention of studying the effects of the amount of coarsening on the graph partition quality and segmentation time. In order to have a fair comparison, all the graphs are coarsened by Heavy Edge matching algorithm, and the partitions are projected without refinement.

Table D.1 shows the Ncut values for all the 10 levels of coarsening for 4-way segmentation.
of the test graphs while Table D.2 shows the results for 8-way segmentation. These results are presented graphically in Figure 4-9. For the graph *eppstein*, no data is available at level 10 for 4-way partition and at level 8, 9, and 10 for 8-way partition. The reason for this is that the graph is coarsened to the extent where the number of nodes of the coarsest graph are less than the required number of partition, $k$. The size of the adjacency matrix, $W$ is less than $k$. Hence, it is not possible to obtain $k$ independent eigenvectors from the adjacency matrix. The multilevel will not work if the graph is coarsened until this extent. The same reason applies for the unavailability of data at some levels for the graph *smallmesh*.

In general, the $Ncut$ value increases with the amount of coarsening applied to the graph. The highest increase in the $Ncut$ value 1413.58% for 4-way partition (tapir) and 2768.59% for 8-way partition (spiral), both at level 10 respectively. However, as seen in Table D.1, there are graphs that experience a decrease in the $Ncut$ value even though there is no refinement applied to them. This is surprising since too much of coarsening may drive the optimal partition for the coarse graph further away from the optimal partition for the original graph.

In terms of the segmentation time, the time required to partition the graph decreases as more coarsening is applied to the graph. Comparing the top and bottom bar charts shown in Figure 4-10, we see that if more partitions are required (higher $k$), the reduction in the running time is less. Like for instance, the improvement in segmentation time for the graph *crack* is about 30% for 4-way partition. However, the reduction in segmentation time for 8-way partition is only about 10%. Note that the time reduction in segmentation for graph bisection is the highest (in the region of 60%, refer to Table A.2).

For *smallmesh*, the segmentation time for 8-way partition at 2 levels is actually higher than the Fiedler method by 1.55%. This means that the time advantage of the multilevel algorithm is lost. However, this is a small graph with only 136 nodes. For a 8-way partition of this graph, the maximum levels of coarsening for this graph is three as further coarsening will cause the graph to have less than eight nodes. Hence, the maximum reduction in time for partitioning this graph into eight segments without refinement is 27.58%.

4.2.2 Partitioning Phase

In the partitioning phase, the coarsest graph is partitioned into $k$ segments either by the recursive bisection or multiclass spectral clustering method. The $Ncut$ value is higher for more partitions since more edges will be cut. Table 4.6 compares the recursive bisection and multiclass spectral clustering in terms of the $Ncut$ value and segmentation time. The time and $Ncut$ value recorded are obtained by partitioning the graph by Fiedler method (without
Figure 4-9: Effects of the number of coarsening levels on the $Ncut$ value for partitioning the graph into four and eight segments ($k = 4$ and $k = 8$). In general, the quality of the partition deteriorates with increasing amount of coarsening.
Effects of coarsening levels execution time ($k=4$)

Effects of coarsening levels execution time ($k=8$)

Figure 4-10: Effects of the number of coarsening levels on execution time for partitioning the graph into four and eight segments ($k = 4$ and $k = 8$). In general, the running time required decreases with increasing amount of coarsening. The reduction in running time for the 8-way partition is less as compared to the 4-way partition.
multilevel implementation).

Table 4.6: Comparison between the $Ncut$ value and segmentation time obtained by recursive bisection and multiclass spectral clustering for $k = 8$. The multiclass spectral clustering method outperforms the recursive bisection method in terms of speed and quality.

<table>
<thead>
<tr>
<th>Graph</th>
<th>$Ncut$</th>
<th>% Difference</th>
<th>Time (second)</th>
<th>% Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Recursive</td>
<td>Multiclass</td>
<td></td>
<td>Recursive</td>
</tr>
<tr>
<td>airfoil1</td>
<td>0.2777</td>
<td>0.2215</td>
<td>-9.74</td>
<td>7.11</td>
</tr>
<tr>
<td>airfoil2</td>
<td>0.0135</td>
<td>0.2287</td>
<td>-17.64</td>
<td>7.58</td>
</tr>
<tr>
<td>airfoil3</td>
<td>0.1394</td>
<td>0.1198</td>
<td>-14.06</td>
<td>56.82</td>
</tr>
<tr>
<td>crack</td>
<td>0.3482</td>
<td>0.3014</td>
<td>-13.44</td>
<td>7.50</td>
</tr>
<tr>
<td>parc</td>
<td>0.2568</td>
<td>0.2485</td>
<td>-3.23</td>
<td>2.95</td>
</tr>
<tr>
<td>parcweb</td>
<td>0.3758</td>
<td>0.2102</td>
<td>-44.07</td>
<td>4.00</td>
</tr>
<tr>
<td>spiral</td>
<td>0.1718</td>
<td>0.1726</td>
<td>0.47</td>
<td>4.25</td>
</tr>
<tr>
<td>tapir</td>
<td>0.4735</td>
<td>0.4316</td>
<td>-8.85</td>
<td>1.57</td>
</tr>
<tr>
<td>eppstein</td>
<td>0.9354</td>
<td>0.8009</td>
<td>-14.38</td>
<td>0.67</td>
</tr>
<tr>
<td>smallmesh</td>
<td>1.8681</td>
<td>1.6852</td>
<td>-9.79</td>
<td>0.22</td>
</tr>
</tbody>
</table>

From the table, we see that the multiclass spectral clustering method partitions the graph faster than the recursive bisection. The different in partitioning time for these two methods is huge. The multiclass spectral clustering method is faster by 65-85% as compared to the recursive bisection method. This is because the multiclass spectral clustering only solve a generalized eigenvalue problem once and discretizes these $k$ eigenvectors into $k$ discrete partitions while the recursive bisection method cut the graph repeatedly into smaller graphs until $k$ partitions are obtained (by solving multiple eigenvalue problems on the subgraphs).

With respect to the segmentation quality, the multiclass method yields better segmentation than the recursive bisection method (except for spiral). In particular, we see that the $Ncut$ value for the graph parcweb is 44.07% lower for the multiclass method. This is because the multiclass spectral clustering optimized the $Ncut$ value simultaneously while the recursive bisection method only optimizes the $Ncut$ value for the bisection at each recursion.

Figure 4-11 shows the comparison between the recursive bisection and multiclass spectral clustering for 2, 4, and 8 partitions (for airfoil1 and tapir). The main disadvantage of the recursive bisection is evident in these two figures. For the graph tapir, the horse-like mesh is
split into two parts: the head and body. Subsequently, the recursive bisection only has the choice of further splitting the head and body into four parts separately while the multiclass method has the freedom of splitting the mesh into four partitions simultaneously. In this case, the horse head is preserved homogenously. Similar observation is made for the graph \textit{airfoil1}. The comparison between these two methods for the other test graphs are shown in Appendix E.

### 4.2.3 Uncoarsening Phase

**Inverse Powering**

Table 4.7 shows the results of inverse powering method for partitioning the test graphs into four partitions. All the projected vectors converged to the Fiedler vectors of $G_0$. The norm shown on column 8 of the table is the matrix norm of the difference between the matrix that contains the first $k$ eigenvectors of $G_0$ and the matrix of the converged vectors by the inverse iteration. We see that all the matrix norm is less than 1e-11.

The method offers time reduction for all the graphs except \textit{crack} and \textit{smallmesh}. The partitioning quality obtained by the Inverse Powering method is the same as partitioning using the Fiedler method.

Table 4.7: Results of inverse powering method for partitioning the test graphs into four partitions. All project vectors converged to the Fiedler vectors of $G_0$. The method offers time reduction for all the graphs except \textit{crack} and \textit{smallmesh}. The partitioning quality obtained by the inverse powering method is the same as partitioning using Fiedler method (without multilevel implementation).

<table>
<thead>
<tr>
<th>Graph</th>
<th>Ncut</th>
<th>Ncut</th>
<th>Time (second)</th>
<th>Time (second)</th>
<th>Time (second)</th>
<th>Time (second)</th>
<th>Norm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fiedler</td>
<td>multilevel</td>
<td>% Diff</td>
<td>Fiedler</td>
<td>multilevel</td>
<td>% Diff</td>
<td>Fiedler</td>
</tr>
<tr>
<td>airfoil1</td>
<td>0.0793</td>
<td>0.0793</td>
<td>0.0000</td>
<td>2.4552</td>
<td>1.5851</td>
<td>-35.4391</td>
<td>1.43E-13</td>
</tr>
<tr>
<td>airfoil2</td>
<td>0.0926</td>
<td>0.0926</td>
<td>0.0000</td>
<td>2.6215</td>
<td>2.0796</td>
<td>-20.6714</td>
<td>7.65E-14</td>
</tr>
<tr>
<td>airfoil3</td>
<td>0.0437</td>
<td>0.0437</td>
<td>0.0000</td>
<td>22.7803</td>
<td>11.2787</td>
<td>-50.4892</td>
<td>2.04E-13</td>
</tr>
<tr>
<td>crack</td>
<td>0.1040</td>
<td>0.1040</td>
<td>0.0000</td>
<td>2.1502</td>
<td>2.8204</td>
<td>31.1692</td>
<td>7.58E-12</td>
</tr>
<tr>
<td>parc</td>
<td>0.0820</td>
<td>0.0820</td>
<td>0.0000</td>
<td>0.9313</td>
<td>0.4036</td>
<td>-56.6627</td>
<td>1.92E-13</td>
</tr>
<tr>
<td>parcweb</td>
<td>0.0700</td>
<td>0.0700</td>
<td>0.0000</td>
<td>1.233</td>
<td>0.7259</td>
<td>-41.1273</td>
<td>7.70E-14</td>
</tr>
<tr>
<td>spiral</td>
<td>0.0447</td>
<td>0.0447</td>
<td>0.0000</td>
<td>1.5092</td>
<td>0.4251</td>
<td>-71.8328</td>
<td>3.33E-13</td>
</tr>
<tr>
<td>tapir</td>
<td>0.0788</td>
<td>0.0788</td>
<td>0.0000</td>
<td>0.4523</td>
<td>0.335</td>
<td>-25.9341</td>
<td>1.10E-12</td>
</tr>
<tr>
<td>eppstein</td>
<td>0.2300</td>
<td>0.2300</td>
<td>0.0000</td>
<td>0.1933</td>
<td>0.1565</td>
<td>-19.0378</td>
<td>2.04E-13</td>
</tr>
<tr>
<td>smallmesh</td>
<td>0.5795</td>
<td>0.5795</td>
<td>0.0000</td>
<td>0.0467</td>
<td>0.069</td>
<td>47.7516</td>
<td>2.40E-15</td>
</tr>
</tbody>
</table>

70
Figure 4-11: Comparison between the recursive bisection and multiclass spectral clustering method for 2, 4, and 8 partitions for graph *airfoil1* and *tapir*. For the recursive bisection method, the method only has the choice of further splitting the subgraphs obtained at the previous recursion while the multiclass method can simultaneously segment the graph into the number of required partitions.
Combination of Inverse Powering with Greedy Local Refinement

Table 4.8 shows the results of the combination of the inverse powering method with greedy local refinement. The greedy local refinement is performed on the partition obtained by the inverse powering method. From the table, we see that the combination method gives lower Ncut values for all graphs except spiral. The Ncut value for eppstein improved by the biggest margin, from 0.2300 decreases to 0.2103, a decrement of 8.57%. The Ncut value of spiral is exactly the same as the Fiedler method. This could be that the partition is already an optimal partition.

However, the segmentation time required for the combination method is higher than partitioning the graphs without the multilevel implementation except for spiral. The segmentation time of the combined method for spiral is still lower than the Fiedler method since the refinement iteration exits immediately with no improvement found. For the rest of the graphs, the refinement iteration terminates when there is no further improvement in the Ncut value. The number of refinement iteration can be capped to have shorter segmentation time at the expense of higher Ncut value.

Table 4.8: Results of the combination of the inverse powering method with greedy local refinement for partitioning the test graphs into four partitions. All graphs except spiral have improvement in the Ncut value, ranging from 1.37 to 8.57%. The Ncut value for spiral is the same as the Fiedler method. This could be that the partition is already optimal. However, the segmentation time of the combination of inverse powering with greedy refinement is longer as compared to the time required for the spectral partition of the original graph.

<table>
<thead>
<tr>
<th>Graph</th>
<th>Ncut</th>
<th>Time (second)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fiedler</td>
<td>multilevel</td>
</tr>
<tr>
<td>airfoil1</td>
<td>0.0793</td>
<td>0.0778</td>
</tr>
<tr>
<td>airfoil2</td>
<td>0.0926</td>
<td>0.0940</td>
</tr>
<tr>
<td>airfoil3</td>
<td>0.0437</td>
<td>0.0431</td>
</tr>
<tr>
<td>crack</td>
<td>0.1040</td>
<td>0.1003</td>
</tr>
<tr>
<td>parc</td>
<td>0.0820</td>
<td>0.0785</td>
</tr>
<tr>
<td>parcweb</td>
<td>0.0700</td>
<td>0.0674</td>
</tr>
<tr>
<td>spiral</td>
<td>0.0447</td>
<td>0.0447</td>
</tr>
<tr>
<td>tapir</td>
<td>0.0788</td>
<td>0.0750</td>
</tr>
<tr>
<td>eppstein</td>
<td>0.2300</td>
<td>0.2103</td>
</tr>
<tr>
<td>smallmesh</td>
<td>0.5795</td>
<td>0.5553</td>
</tr>
</tbody>
</table>
Chapter 5

Experimental Results for Image Segmentation

In this chapter, I present the image segmentation results obtained by the multilevel clustering algorithm described in Chapter 3. Various synthetic and natural images are used as test images for multilevel image segmentation algorithm. These images were obtained from Google's image search and UC Berkeley Hand Segmentation Database [13]. In the hand segmentation database, a total of 12,000 hand-labeled segmentations of 1,000 Corel dataset images from 30 human subjects were made available online\(^1\).

While the normalized cut (\(N\text{cut}\)) value provides an indication of the quality of a partition, finding the optimal \(N\text{cut}\) value does not automatically translates to the best possible partition. The success of spectral image partition lies in building an adjacency matrix which reflects the likelihood of two pixels belongs to a group. In low level image segmentation, which is the focus in this thesis, only the intensity, distance, and contour information of an image are used to weight the edges. Choosing the optimal parameters for the weighting scheme directly affects the segmentation quality. Hence, segmenting natural images with only the low level information is challenging and difficult.

This chapter begins with the analysis of four weighting schemes used in building the adjacency matrix. Subsequently, I present the breakdown of the findings in the three phases: coarsening, partitioning, and uncoarsening. An image can be partitioned into \(k\) segments either by recursive bisection or multiclass spectral clustering. The advantage and disadvantage of these two methods are discussed. The segmentation results for natural images (from the hand segmentation database) are shown at the end of this chapter.

\(^1\)available online at http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsd5/
5.1 Weighting Function for Adjacency Matrix

Shown in the second column of Figure 5-1, 5-2, and 5-3 are the affinity plots for the images of 1) a white square in black background, 2) bird, and 3) MIT dome using the four different weighting schemes (linear, exponential, Gaussian and combined cue). Each point in the contour plot represents the degree of the corresponding pixel in the image, where $\text{degree}(i) = \sum_j W_{ij}$. Each row of the adjacency matrix, $W(i,:)$, is reshaped into the size of the image and summed to produce these affinity plots. The higher the degree value, the more strongly linked the node is to its neighboring nodes. For the nodes along the edges of an object, by construction, the degree value will be less. The original images are shown on the top left corner of their respective figures.

The synthetic image of a white box against black background shown in Figure 5-1 has pixel value 255 at the white box region and pixel value 0 at the black background. The affinity plots for all the four weighting schemes shown in the second column are similar. The links along the boundaries between the white box and the background is the weakest since the difference in pixel intensity is the largest along these boundaries. For this relatively simple image, all the four schemes were able to segment the image correctly. The image is partitioned into two segments with the white box (shown in either red or green box in column 3 of Figure 5-1) separated from the background.

From the affinity plots for an image of a bird shown in Figure 5-2, the outline of the bird can be clearly seen for all the four weighting schemes. The linear weighting scheme exhibits the lowest contrast in the edges as compared to the other three schemes. This is because the decay rate of the edge weight is the lowest among all the four schemes. On the other hand, the difference in the degree value is largest along the edge of the bird for the combined cue weighting scheme as compared to the linear, exponential and Gaussian schemes.

The segmentation results for the bird are shown in the second column of Figure 5-2. From the results, we see that the linear scheme is not adequate in segmenting the bird out from its background. The best segmentation is obtained by the combined cue scheme. Both the exponential and Gaussian weighting schemes failed to segment the bird homogenously.

Figure 5-3 shows the affinity plot for an image of the MIT Dome. Once again, from the segmentation results, we see that the linear weighting scheme yields the worst partition while the combined cue scheme yields the best partition. The combined cue weighting scheme could segment the two trees on the left and right, and the field correctly. Both the exponential and Gaussian schemes exhibit similar segmentation quality.

From these results, we see that the linear scheme is not adequate in segmenting natural
images. This scheme should only be used for relatively easy image such that the black-white square image. The segmentations given by the exponential and Gaussian weighting schemes are better than the linear scheme, but worst than the combined cue scheme. Hence, the combined cue weighting scheme is a better scheme as compared to the other three schemes. In this thesis, the combined cue weighting scheme is used for all experiments.

5.2 Coarsening Phase

In the coarsening phase, the graph derived from an image is shrunk into a smaller graph by collapsing the vertices and edges. Table 5.1 summarizes the size of the graph of ten test images for five levels of coarsening. The test images are resized from their original sizes to a size of about 200x200. After five levels of coarsening, the size of the graphs is around 10 to 20% of the size of $G_0$.

The graphs for the test images listed on Table 5.1 are coarsened from 1 to 10 levels to study the effect of coarsening to the $Ncut$ value and partitioning time. In all experiments for the multilevel image segmentation, the graphs are coarsened by Heavy Edge matching. As the matching algorithm is randomized in nature, I choose to implement the coarsening process in a fixed order. The nodes are matched in order of decreasing edge weight. The two nodes with the stronger link (higher edge weight) are matched first.

The $Ncut$ value and segmentation time for various levels of coarsening are summarized in Table 5.2 and 5.3, respectively. The graphical representations of these results are shown in Figure 5-4 and 5-5. No refinement is applied to the projected partition. In general, the $Ncut$ value increases while the execution time decreases for more coarsening levels.

From Table 5.2, we see that the $Ncut$ value for the graph of the image flower stays constant even after one level of coarsening. However, the $Ncut$ value sharply increased by 525% for two levels of coarsening. Furthermore, after ten levels of coarsening, the $Ncut$ value increased by a whopping 2225%.

Interestingly, in contrast to the results of multilevel graph partitioning shown in Section 4.1.1, the multilevel image segmentation without refinement step can yield a $Ncut$ value lower than segmenting the original image (refer to the entries highlighted in bold font in Table 5.2). The $Ncut$ value for the graph of the image butterfly decreased by 54.55% and 45.45% after two and three levels of coarsening, respectively. For the image panther, the $Ncut$ value improves by 10 to 20% even though the graph is coarsened for six to ten levels of coarsening.
Figure 5-1: The top left corner shows a synthetic image with a white square in the center against black background. The affinity plots for the four different weighting schemes are shown in column 2. The segmented results for the corresponding weighting schemes are shown in column 3. For this relatively simple image, all four weighting schemes yield the correct segmentation.
Figure 5-2: The top left corner shows an image of a bird. The affinity plots for the four different weighting schemes are shown in column 2. The segmented results for the corresponding weighting schemes are shown in column 3. The combined cue weighting scheme yields the best segmentation among the four schemes. For the exponential and Gaussian scheme, the image is segmented along the weak edges. However, these two schemes failed to segment the bird homogenously as compared to the combined cue scheme. The linear weighting scheme yields the worst partition as the edge weight has slow a rate of decay.
Figure 5-3: The top left corner shows an image of the MIT dome. The affinity plots for the four different weighting schemes are shown in column 2. The segmented results for the corresponding weighting schemes are shown in column 3. The best partition is obtained by the combined cue weighting scheme. The two trees on the left and right and the field are correctly segmented. The partition quality given by the exponential and Gaussian schemes is similar, but worse than the combined cue scheme. The linear weighting scheme is not adequate in segmenting the image.
Table 5.1: Size of the graph, $|V_i|$ for 5 levels of coarsening.

| Graph       | Image Size | $|V_0|$ | Level 1 | Level 2 | Level 3 | Level 4 | Level 5 | %Fraction |
|-------------|------------|--------|---------|---------|---------|---------|---------|-----------|
| Bird        | 200x200    | 40000  | 27306   | 19336   | 13349   | 8687    | 5407    | 13.52     |
| Panther     | 200x300    | 60000  | 40750   | 25432   | 15663   | 9566    | 5882    | 9.80      |
| MIT Dome    | 200x300    | 60000  | 41409   | 28467   | 19272   | 12368   | 7677    | 12.80     |
| Baby        | 200x204    | 40800  | 29699   | 22704   | 16928   | 12260   | 8604    | 21.09     |
| Flower      | 200x267    | 53400  | 36728   | 25572   | 17505   | 11355   | 7059    | 13.22     |
| Grass       | 200x300    | 60000  | 41010   | 25381   | 15659   | 9587    | 5944    | 9.91      |
| Taj Mahal   | 200x286    | 57200  | 39391   | 27434   | 18686   | 12178   | 7672    | 13.41     |
| Landscape   | 200x267    | 53600  | 36565   | 25192   | 17074   | 11029   | 6881    | 12.84     |
| Butterfly   | 200x267    | 53400  | 36693   | 23889   | 15212   | 9419    | 5820    | 10.90     |
| Skater      | 200x302    | 60400  | 41560   | 29282   | 20282   | 13260   | 8319    | 13.77     |

From Table 5.3, we see that the reduction in segmentation time is more than 90% for four levels of coarsening onwards. The segmentation time include the time needed for all the three phase of multilevel clustering: coarsening, partitioning, and uncoarsening. The marginal gain in reduction time reduces as the graph is coarsened for more than five levels. Coarsening the graph further does not provide significant reduction in the segmentation time. The time required to segment the original image (without multilevel algorithm) is shown on column 2 (Fiedler) of the table.
Table 5.2: \textit{Ncut} Values for various levels of coarsening (no refinement)

<table>
<thead>
<tr>
<th>Graph</th>
<th>Fiedler</th>
<th>%Δ in \textit{Ncut} for Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Bird</td>
<td>0.0009</td>
<td>0.00</td>
</tr>
<tr>
<td>Panther</td>
<td>0.0202</td>
<td>\textbf{-12.87}</td>
</tr>
<tr>
<td>MIT Dome</td>
<td>0.0016</td>
<td>6.25</td>
</tr>
<tr>
<td>Baby</td>
<td>0.0007</td>
<td>\textbf{-57.14}</td>
</tr>
<tr>
<td>Flower</td>
<td>0.0008</td>
<td>0.00</td>
</tr>
<tr>
<td>Grass</td>
<td>0.0037</td>
<td>13.51</td>
</tr>
<tr>
<td>Taj Mahal</td>
<td>0.0111</td>
<td>\textbf{-1.80}</td>
</tr>
<tr>
<td>Landscape</td>
<td>0.0011</td>
<td>9.09</td>
</tr>
<tr>
<td>Butterfly</td>
<td>0.0192</td>
<td>\textbf{-45.31}</td>
</tr>
<tr>
<td>Skater</td>
<td>0.0036</td>
<td>11.11</td>
</tr>
</tbody>
</table>

Table 5.3: Execution time for various levels of coarsening (no refinement)

<table>
<thead>
<tr>
<th>Graph</th>
<th>Fiedler</th>
<th>%Δ in execution time for Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Bird</td>
<td>158.72</td>
<td>-41.05</td>
</tr>
<tr>
<td>Panther</td>
<td>212.97</td>
<td>-41.93</td>
</tr>
<tr>
<td>MIT Dome</td>
<td>234.27</td>
<td>-39.27</td>
</tr>
<tr>
<td>Baby</td>
<td>336.68</td>
<td>-30.86</td>
</tr>
<tr>
<td>Flower</td>
<td>294.30</td>
<td>-45.48</td>
</tr>
<tr>
<td>Taj Mahal</td>
<td>152.37</td>
<td>-35.82</td>
</tr>
<tr>
<td>Landscape</td>
<td>286.41</td>
<td>-45.47</td>
</tr>
<tr>
<td>Butterfly</td>
<td>142.45</td>
<td>-39.40</td>
</tr>
<tr>
<td>Skater</td>
<td>188.21</td>
<td>-42.89</td>
</tr>
</tbody>
</table>
Figure 5-4: Effects of the number of coarsening levels on the $N_{cut}$ value. The $N_{cut}$ value increases with the number of coarsening level.

Figure 5-5: Effects of the number of coarsening levels on the execution time. The multilevel partitioning time is significantly shorter than segmenting the original image. The execution time decreases as the number of coarsening level increases.
5.2.1 Using Ncut value as Measure of the Quality of Image Segmentation

In spectral image segmentation, the first $k$ smallest eigenvectors obtained by solving a generalized eigenvalue system, $(D - W)x = \lambda Dx$, are discretized to achieve $k$ partitions. Shi and Malik [16] proved that the second smallest eigenvector is the relaxed real valued solution to the normalized cut problem. Hence, spectral image segmentation yields a partition which has near optimal $Ncut$ value.

However, lower $Ncut$ value does not necessarily mean better partition quality. From the structure of the normalized cut formulation, finding a segmentation which cut along the weakest links will yield the lowest $Ncut$ value. In order to get a segmentation which is close to human hand segmentation, the edge weights have to be weighted such that the weight along the desired segmentation is the weakest.

For humans, a better partition would be to preserve the salient part of an object [15]. For example, in Figure 5-2, the combined cue weighting scheme successfully segment the bird homogenously out from its background, while the other schemes segment the image along other weak edges. The low level segmentation cues used in building the adjacency matrix alone are not sufficient to ensure the edge weight along salient object is the weakest. Therefore, the best partition for humans may not have the lowest $Ncut$ value.

Shown in Figure 5-6a is an image of a landscape. Figure 5-6b shows the segmentation obtained by Fiedler method (segmenting the original image) while Figure 5-6c and 5-6d shows the segmentation obtained by the multilevel algorithm after two and three levels of coarsening, respectively. From the figure, three observations can be made:

1. First, segmenting the original image by spectral clustering method (Fiedler) does not give the minimum $Ncut$ value. This is expected since the spectral method is the relaxed real valued solution the normalized cut problem.

2. Second, the multilevel segmentation algorithm can yield a partition with lower $Ncut$ value even though no refinement is applied to the projected partition. For this image, the $Ncut$ value is reduced by 54.55 and 45.45% after two and three levels of coarsening, respectively.

3. Third, the best segmentation for this image (to humans) is neither obtained by segmenting the original image nor finding the partition with the lowest $Ncut$ value. The segmentation closest to human segmentation is shown in Figure 5-6d with a $Ncut$
value of 0.0005 obtained with two levels of coarsening. Hence, the Ncut value is not the absolute gauge for the image segmentation quality. The partition which gives the lowest Ncut value does not mean best partition.

In addition, for this image, the multilevel segmentation without refinement finds the best partition with significantly shorter time. Segmenting the original image of size 200x267 pixels requires 286.41 second. In the multilevel scheme, the segmentation time for two and three levels of coarsening are 78.15 and 36.47 second, respectively. Therefore, the multilevel algorithm finds the best partition (at three levels of coarsening), and at the same time offers a reduction of 87.27% in the segmentation time.

Shown in Figure 5-7a is an image with two flowers. For this image, the lowest Ncut value is obtained by the Fiedler method (segmenting the original image). After two levels of coarsening, the Ncut value increased steeply from 0.0008 to 0.0050. However, even though the Ncut value increased by 525%, the segmentation quality is the same as the one obtained by Fiedler (refer to Figure 5-7b and 5-7c). In both cases, the two flowers were segmented homogenously from its background. Again, the Ncut failed to give a correct interpretation of the quality of segmentation. Nevertheless, after ten levels of coarsening, the Ncut value is increased by 2225%, from 0.0008 to 0.0186. Only one flower is segmented homogenously while the other flower is grouped together with one portion of the background.

Therefore, finding a partition which has the lowest Ncut value does not automatically mean the best partition. The edge weights have to be weighted to reflect the likelihood of two pixels belong to the same object. While the Ncut value is not the ultimate yardstick to the segmentation quality of an image, the Ncut value nonetheless still provides a good indication on the segmentation quality. Refer to Appendix F for the segmentation of the test images from 1 to 10 levels of coarsening. Note that the segmentation obtained by the first few levels of coarsening is similar to the segmentation obtained by the Fiedler method.

5.3 Partitioning Phase

In the partitioning phase, the coarsest graph, $G_m$, is partitioned into $k$ segments either by the recursive bisection or multiclass spectral clustering method. For the recursive bisection, the coarsest graph is first partitioned into two segments. These two segments are then further partitioned into four partitions. The process continues until $k$ partitions are achieved. For the multiclass spectral clustering, a generalized eigenvalue problem is solved on the coarsest
Figure 5-6: Three observations can be made here: First, Fiedler method (multiclass spectral clustering) does not give the minimum $N_{cut}$ value; Second, even after two levels of coarsening, the lowest $N_{cut}$ value is 54.55% lower than the Fiedler’s $N_{cut}$ value; Third, the best partition to human eye is obtained not by Fiedler nor the lowest $N_{cut}$ value, but at $N_{cut} = 0.0006$. The time for segmenting the original image (of size 200x267) is 286.41 second. The segmentation time for two and three levels of coarsening are 78.15 and 36.47 second, respectively. In this example, the multilevel algorithm finds the best partition with 87.27% reduction in segmentation time.
Figure 5-7: For this image, the lowest $Ncut$ value is obtained by the Fiedler method (segmenting the original image). However, even though the $Ncut$ value at two levels of coarsening increased by 525%, the segmentation quality is the same as the one obtained by Fiedler. In both cases, the two flowers were segmented homogenously from its background (compare (b) and (c)). At ten levels of coarsening, the $Ncut$ value is increased by a massive 2225%. Only one flower is segmented homogenously while the other flower is grouped with one portion of the background.
graph to obtain its first $k$ smallest eigenvectors. These $k$ eigenvectors are then discretized into $k$ partitions by the rounding method.

Table 5.4 shows the comparison between the recursive bisection and multiclass spectral clustering in terms of the $Ncut$ value and segmentation time. As we are comparing the advantage and disadvantages of these two partitioning methods, the original image is segmented without the multilevel algorithm. All the test images are resized to about 100x100 pixels and the number of partition is set to eight ($k = 8$).

From this table, we see that the recursive bisection method yields a better $Ncut$ value than the multiclass spectral clustering except for three images: Baby, Taj Mahal, and Skater. This is quite surprising since the recursive bisection only optimizes the $Ncut$ value of the bipartition at each recursion stage while the multiclass spectral clustering optimizes the $Ncut$ value for the $k$ partitions simultaneously.

The recursive bisection method takes much longer time than the multiclass spectral clustering method to partition the graph into $k$ segments. From Table 5.4, the segmentation time for multiclass spectral method is more than 90% faster than the recursive bisection method. This is because the recursive bisection method needs to partition the graph repeatedly, while multiclass only solve for the generalized eigenvalue system once and discretize the eigenvectors to obtain $k$ partitions. The recursive bisection method is prohibitively expensive for segmenting images of size above 200x200 pixels. For example, the segmentation time (without multilevel) for the image panther and MIT dome of size 200x300 pixels takes 1729.19 and 3729.15 seconds, respectively.

The main disadvantage of the recursive bisection method is that the method is unable to segment the objects in an image simultaneously. The recursive bisection method continuously split the graph into smaller segments even though the subgraphs may contain salient objects. Shown in Appendix G are two flow charts for the recursive bisection steps for segmenting the image MIT Dome and Heart into eight partitions. The segmentation results obtained by the recursive bisection and multiclass spectral clustering method for these two images are presented in Figure 5-8.

From Figure G-1 shown on page 139, we see that the method found the two trees on the left and right of the dome at the second level of recursion. At the subsequent level of recursion, the two subgraphs which contain the two trees together with other subgraphs are further segmented into smaller parts until the required numbers of partition are obtained.

The same problem also occurs when segmenting the image Heart (refer to Figure G-2 on page 140). There are multiple love-shaped objects in this image. The recursive bisection
Table 5.4: Comparison between the Ncut value and segmentation time obtained by recursive bisection and multiclass spectral clustering for \( k = 8 \). The test images are resized to about 100x100 in size.

<table>
<thead>
<tr>
<th>Graph</th>
<th>( Ncut )</th>
<th>( % ) Difference</th>
<th>Time (second)</th>
<th>( % ) Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Recursive</td>
<td>Multiclass</td>
<td>Recursive</td>
<td>Multiclass</td>
</tr>
<tr>
<td>Bird</td>
<td>0.1123</td>
<td>0.1156</td>
<td>65.47</td>
<td>3.81</td>
</tr>
<tr>
<td>Panther</td>
<td>0.0259</td>
<td>0.0409</td>
<td>188.58</td>
<td>14.50</td>
</tr>
<tr>
<td>MIT Dome</td>
<td>0.0274</td>
<td>0.0620</td>
<td>230.81</td>
<td>9.25</td>
</tr>
<tr>
<td>Baby</td>
<td>1.0353</td>
<td>0.0733</td>
<td>112.33</td>
<td>4.72</td>
</tr>
<tr>
<td>Flower</td>
<td>0.0036</td>
<td>0.0121</td>
<td>170.80</td>
<td>11.11</td>
</tr>
<tr>
<td>Grass</td>
<td>0.1108</td>
<td>0.1563</td>
<td>101.99</td>
<td>6.21</td>
</tr>
<tr>
<td>Taj Mahal</td>
<td>0.2867</td>
<td>0.0205</td>
<td>160.51</td>
<td>10.75</td>
</tr>
<tr>
<td>Landscape</td>
<td>0.0212</td>
<td>0.0915</td>
<td>135.41</td>
<td>7.51</td>
</tr>
<tr>
<td>Butterfly</td>
<td>0.0310</td>
<td>0.0454</td>
<td>103.09</td>
<td>10.01</td>
</tr>
<tr>
<td>Skater</td>
<td>0.1306</td>
<td>0.0421</td>
<td>194.33</td>
<td>10.39</td>
</tr>
</tbody>
</table>

method finds more and more of these objects with more levels of recursion. However, once the objects are found, they are further bisected into smaller parts at the next recursion.

Hence, multiclass spectral clustering is a better method to segment an image into \( k \) partitions. The method is able to simultaneously segment the objects in the image provided that the edge weights along the object edges are weak. In this thesis, all experiments for image segmentation are performed using the multiclass spectral clustering method to partition the coarsest graph, \( G_m \) into \( k \) partitions. These partitions are then projected back through the levels to the original graph \( G_0 \).

### 5.4 Uncoarsening Phase

In the uncoarsening phase, the base partition of the coarsest graph is projected back to the graph of the original image. As the graph is uncoarsened, the quality of the projected partition can be improved by applying refinement steps. For the multilevel image segmentation, we have seen examples where the projected partition without refinement offers a better segmentation quality albeit having higher \( Ncut \) values. However, there are also images where segmenting the original image gives the best segmentation.

In this section, I present the results of the two refinement scheme: inverse powering
Figure 5-8: Comparison of the segmentation results obtained by the recursive bisection and multiclass spectral clustering method for two images: *MIT Dome* and *Heart*. The recursive bisection method failed to segment the object in the image homogenously. The method keeps bisecting the graph into smaller parts, even though the subgraph contains the salient object. On the contrary, the multilevel spectral clustering method is able to segment the objects in the image homogenously.
and greedy refinement method. These two refinement methods are applied to the projected partition with the intention of achieving the same or better segmentation quality (for the multilevel method) as partitioning the original graph.

5.4.1 Inverse Powering Method

The inverse powering method is not applicable if the recursive bisection is used as the base partitioning method. The inverse powering method makes use of the eigenvectors obtained by solving the generalized eigenvalue system on the coarsest graph as starting vectors for the inverse iteration. In the recursive bisection method, the coarsest graph is partitioned repeatedly bisected until the graph is partitioned into $k$ segments. Therefore, the first $k$ smallest eigenvectors of the coarsest graph is not available for the inverse powering method.

On the other hand, the inverse powering method can be used as a refinement scheme when the multiclass spectral clustering method is used at the partitioning phase. The first $k$ smallest eigenvectors of the coarsest graph are projected to the finer graphs $G_{m-1}, G_{m-2}$, back to $G_0$ by injection. These $k$ eigenvectors are normalized at each uncoarsening step. These projected eigenvectors and its corresponding Rayleigh quotients are used as the starting eigenvectors and eigenvalues for the inverse powering method.

However, in the $k$-way image segmentation, the eigenvectors may converge to the higher or lower eigenvectors of the original graph $G_0$. If such a case occurs, the image will have less than the required $k$ segments. Hence, the inverse iteration are repeated until all the eigenvectors converge to the first $k$ smallest eigenvectors of the graph derived from the original image. Instead of using the Rayleigh quotient of the corresponding starting vector as the starting eigenvalue, the eigenvalue of the previous eigenvector is used. If the vector converges to the lower eigenvectors, the inverse iteration is repeated with a higher starting eigenvalue (between the previous eigenvalue and the Rayleigh quotient).

Table 5.5 summarizes the segmentation times and $Ncut$ values for segmenting the image bird by three schemes: 1.) partitioning the original image with multiclass spectral clustering; 2.) multilevel segmentation without refinement; 3.) and multilevel segmentation with the inverse powering method. Three levels of coarsening are applied to both of the multilevel segmentation schemes. In this experiment, the image is resized to sizes of 50x50, 100x100, 200x200, 300x300, and 400x400 pixels. The time efficiency and partition quality of these three schemes at these image sizes are investigated.

From Table 5.5, we see that the $Ncut$ value for the multilevel segmentation without refinement (projection only) is higher than the other two schemes. In contrast, the $Ncut$
value is the same for segmenting the original image and multilevel segmentation with the inverse powering method. The inverse powering method ensures that the segmentation quality obtained by the multilevel partitioning method is identical to the quality obtained by segmenting the original image (without multilevel segmentation).

In terms of the segmentation time, the multilevel segmentation without refinement is significantly faster than the other two schemes. The reduction in the segmentation time is more than 75% for three levels of coarsening. The times taken for segmenting the image at various image sizes are presented graphically in Figure 5-9. For the multilevel segmentation with the inverse powering method, the segmentation times are shorter than segmenting the original image except for the image of size 50x50 pixels.

The multilevel time advantage is lost for segmenting small images since the inverse iteration is repeated until all the eigenvectors converge to the first \( k \) smallest eigenvectors of \( G_0 \). However, for larger graphs, the time advantage is obvious. The larger the original image size, the greater the reduction in the time needed for segmenting the image when using the inverse powering method. Hence, when segmenting large images using the multilevel algorithm, the inverse powering method can be used to obtain identical segmentation quality as partitioning the original images (without the multilevel implementation), and at the same time enjoy a reduction in the segmentation time.

Figure 5-10 shows the comparison of the segmentation results for the image baby obtained by the three schemes used in Table 5.5. Both the multilevel schemes apply 10 levels of coarsening to the original graph. In (c), we see that the multilevel algorithm without refinement segments the object on the top right corner. However, if the inverse powering method is
used in the uncoarsening phase, the segmentation obtained is identical to partitioning the original image without multilevel (compare (b) and (d)). Partitioning the original image without multilevel implementation requires 23.19 second while the multilevel segmentation with the inverse powering method only requires 4.14 second.

Shown in Figure 5-11 is another example where the multilevel segmentation with the inverse powering method used as refinement gives the same segmentation quality (same \(Ncut\) value) as partitioning the original image and offer a shorter segmentation time. For this image, partitioning the original without multilevel requires 38.97 second while the multilevel with the inverse powering method only requires 23.44 second. The projected segmentation obtained by the multilevel segmentation with no refinement is shown in (c).

Figure 5-9: Comparison of segmentation time for three schemes: partitioning the original image with multiclass spectral clustering; multilevel segmentation without refinement; and multilevel segmentation with inverse powering method. The multilevel segmentation without refinement (projection only) is significantly faster than the other two schemes. The multilevel level segmentation with inverse powering method is faster than the time required to segment the original image by multiclass spectral clustering. As the graph size increases, the time advantages of these two multilevel schemes are more evident.
Figure 5-10: Multilevel segmentation for the image baby with 10 levels of coarsening and the inverse powering as refinement method. The segmentation quality obtained by the inverse powering method (refer to (d)) is identical to partitioning the original image without multilevel (refer to (b)). Partitioning the original without multilevel requires 23.19 second while the multilevel with inverse powering only requires 4.14 second.

Figure 5-11: Multilevel segmentation for the image bride with 10 levels of coarsening and the inverse powering as refinement method. The segmentation quality obtained by the inverse powering method (refer to (d)) is identical to partitioning the original image without multilevel (refer to (b)). Partitioning the original without multilevel requires 38.97 second while the multilevel with inverse powering only requires 23.44 second.
5.4.2 Greedy Local Refinement

Since finding a partition with the lowest $Ncut$ value does not translate into the best segmentation, applying the greedy local refinement algorithm by swapping points between partitions as the graph is uncoarsened is not necessary. In the greedy local refinement algorithm, finding the boundary points is the most computationally expensive part of the algorithm. The graphs derived from images are relatively large, thus, the time advantage of multilevel segmentation could be lost if the greedy local refinement is used. Furthermore, applying the greedy refinement algorithm only improves the smoothness of the edges.

Shown in Figure 5-12a is an image of a panther. The $Ncut$ value for segmenting the image (100x150 pixels) without the multilevel implementation is 0.0239. The segmentation is shown in Figure 5-12b. For the multilevel segmentation without refinement (projection only), coarsening the graph for three levels itself already yields an improvement of 28.45% in the $Ncut$ value over the Fiedler method. If the greedy refinement method is used to refine the projected partition as the graph is being uncoarsened, the $Ncut$ value is further improved from 0.0171 to 0.0126. The refinement method improves the $Ncut$ by 47.28% over the $Ncut$ value given by the Fiedler method. However, even though the $Ncut$ value is reduced by a big margin, the two segmentations obtained with and without refinements are indeed similar (compare (c) and (d)).

For this image, the segmentation time for Fiedler is 28.39 second while the multilevel segmentation without refinement only takes 4.18 second. On the other hand, the segmentation time for the multilevel segmentation with the greedy refinement algorithm is 1497.92 second. The greedy refinement terminates when there is no further improvement can be found in the segmentation. Considering that the partition qualities with or without the greedy local refinement algorithm is similar, there is no motivation to apply this refinement algorithm. Nonetheless, the greedy refinement execution time can be lowered by controlling the maximum number of the refinement iterations.

Shown in Figure 5-13 is another example where applying the greedy local refinement method improves the $Ncut$ value, at the expense of running time, but the segmentations with and without refinement are similar. The image of a Koala bear (shown in Figure 5-13a) is partitioned into eight segments. The $Ncut$ value obtained by segmenting the image using multiclass spectral clustering is 0.0567. The projection-only multilevel segmentation for three levels of coarsening gives a partition with a lower $Ncut$ value of 0.0449. The $Ncut$ value is further reduced from 0.0449 to 0.0280 by applying the greedy local refinement algorithm. Again, we see that the segmentations with and without refinement are similar.
Figure 5-12: The multilevel segmentation of the image panther without refinement (projection only) itself already yields an improvement of 28.45% in the $N_{cut}$ value over the Fiedler method. When the greedy local refinement algorithm is applied to the projected partition as the graph is being uncoarsened, the $N_{cut}$ value is further improved from 0.0171 to 0.0126, an improvement of 47.28% as compared to the $N_{cut}$ value given by the Fiedler method. However, even though the $N_{cut}$ value is reduced by a big margin, the two segmentations obtained with and without refinements are indeed similar (compare (c) and (d)).
Figure 5-13: The multilevel segmentation of the image Koala Bear without refinement (projection only) itself already yields an improvement of 20.81% in the $N_{cut}$ value over the Fiedler method. When the greedy local refinement algorithm is applied to the projected partition as the graph is being uncoarsened, the $N_{cut}$ value is further improved from 0.0449 to 0.0280, an improvement of 50.61% as compared to the $N_{cut}$ value given by the Fiedler method. However, even though the $N_{cut}$ value is reduced by a big margin, the two segmentations obtained with and without refinements are indeed similar (compare (c) and (d)).
5.5 Segmentation Results

In this section, I present the results of the multilevel image segmentation algorithm. Here, the segmentation results obtained by the Fiedler method (spectral partition on the original image) and multilevel method without refinement are compared. Since the segmentation results obtained by the inverse powering are the same as partitioning the original image by the spectral method, the results are not presented.

Figure 5-14 and 5-15 show the segmentation results of 48 natural images from the UC Berkeley hand segmentation database [13]. Shown on the first column are the original images. The segmentation by the Fiedler method and multilevel method are shown on column 2 and 3, respectively. From the segmentation results shown in these figures, we see that the multilevel algorithm is able to give similar or better segmentation results than the Fiedler method.

The multilevel method has less tendency of cutting through the objects, thus giving better segmentation quality. This is because, as the graph is coarsened, closely related pixels with high edge weight are collapsed to form a supernode in the coarser graph. For example, for the image shown on the first row of Figure 5-14 (page 97), the Fielder method segments the sky into two parts. On the other hand, the multilevel algorithm segments the moon correctly instead of splitting the sky into two partitions. For the images from row two to six of Figure 5-14, the multilevel algorithm segments the animals (chicks, polar bear, crocodile, and penguin) homogenously while the Fielder method failed to do so.

Table H.1 (on page 142, Appendix H) shows the segmentation parameters, time and $N_{cut}$ value. From the table, we see that the multilevel algorithm offers a reduction in the segmentation time as high as 94.93% as compared to the Fiedler method. Furthermore, the $N_{cut}$ value obtained by the Fiedler method is higher for some images as compared to the $N_{cut}$ value obtained by the multilevel algorithm.

In short, the multilevel image segmentation without refinement gives similar or better partition quality as compared to the Fiedler method, and at the same time, offer significant reduction in the segmentation time.
Continued on the next page
Continued on the next page
<table>
<thead>
<tr>
<th>Original Image</th>
<th>Fiedler</th>
<th>Multilevel</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="159001" /></td>
<td><img src="image2.png" alt="Fiedler" /></td>
<td><img src="image3.png" alt="Multilevel" /></td>
</tr>
<tr>
<td><img src="image4.png" alt="188005" /></td>
<td><img src="image5.png" alt="Fiedler" /></td>
<td><img src="image6.png" alt="Multilevel" /></td>
</tr>
<tr>
<td><img src="image7.png" alt="24063" /></td>
<td><img src="image8.png" alt="Fiedler" /></td>
<td><img src="image9.png" alt="Multilevel" /></td>
</tr>
<tr>
<td><img src="image10.png" alt="293029" /></td>
<td><img src="image11.png" alt="Fiedler" /></td>
<td><img src="image12.png" alt="Multilevel" /></td>
</tr>
<tr>
<td><img src="image13.png" alt="159029" /></td>
<td><img src="image14.png" alt="Fiedler" /></td>
<td><img src="image15.png" alt="Multilevel" /></td>
</tr>
<tr>
<td><img src="image16.png" alt="42044" /></td>
<td><img src="image17.png" alt="Fiedler" /></td>
<td><img src="image18.png" alt="Multilevel" /></td>
</tr>
</tbody>
</table>

Continued on the next page
Continued on the next page
Figure 5-14: First Column: Original Image, Second column: Fiedler segmentation, Third column: Multilevel segmentation. The multilevel algorithm requires shorter segmentation time and offer similar, if not better, segmentation quality as compared to the Fiedler method.
Continued on the next page
Figure 5-15: The multilevel algorithm requires shorter segmentation time and offer similar, if not better, segmentation quality as compared to the Fielder method.
Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, I develop a multilevel graph partitioning scheme that incorporates the inverse powering method with greedy local refinement. With the combination of the inverse powering method and greedy local refinement as the refinement scheme, the partition quality obtained by the multilevel algorithm is equivalent to or better than the spectral partition of the original graph.

In addition, I also presented a scheme to construct the adjacency matrix, \( W \) and degree matrix, \( D \) for the coarse graphs. Due to the the special structure of the Coarsening Matrix, \( CM \), which stores the matching information for coarsening a graph from \( G_i \) to \( G_{i+1} \), the affinity matrices of the coarse graphs can be constructed by the matrix multiplication of \( W_{\text{coarse}} = CM^T \times W_{\text{fine}} \times CM \). With this structure, the affinity matrices of the coarse graphs can be constructed efficiently in Matlab.

The proposed multilevel algorithm is applied to graph partition and image segmentation to test its performance and efficiency. For graph partition, I first tested the multilevel level algorithm for two-way graph partition. In the graph bisection, we have seen that the multilevel algorithm offers significant reduction in segmentation time and at the same time gives the same or better partition quality than solving a generalized eigenvalue problem on the original graph.

For the \( k \)-way graph partition, the starting eigenvector for the inverse iteration may converge to the higher or lower eigenvector of \( G_0 \). If such situation occurs, the number of partition will be \( k \) less the number of independent eigenvectors obtained by the inverse iteration. Hence, to ensure that the inverse powering method converges the starting vectors
to the first $k$ smallest eigenvectors of $G_0$, the eigenvalue of the previous eigenvector will be used as the starting eigenvalue for the inverse iteration. If the vector converges to the lower eigenvectors, the inverse iteration is repeated with a higher starting eigenvalue between the previous eigenvalue and the Rayleigh quotient of the starting vector.

Since the inverse iteration has to be repeated until the first $k$ smallest eigenvector is obtained, the computational time is higher. In the experiments for $k$-way graph partition, we see that the multilevel method with the inverse powering method loses the time advantage for small graphs. However, the time reduction for large graph is obvious. The larger the graph, the greater the reduction in the segmentation time.

The advantages and disadvantages of using the recursive bisection and multiclass spectral clustering method in the partitioning phase are also investigated. The multiclass spectral clustering is a better $k$-way partitioning method as compared to the recursive bisection method. The former only solve for a generalized eigenvalue system once while the latter has to solve multiple eigenvalue problem on the subgraphs at each recursion. Hence, the recursive bisection is slower than the multiclass method. Furthermore, the multiclass method optimizes the $Ncut$ value of the $k$ partitions simultaneously while the recursive bisection only optimizes the $Ncut$ value of the bipartition at each recursion.

For image segmentation, I analyzed four edge weighting schemes for building the image affinity matrix: linear, exponential, Gaussian, and combined cue scheme. Of the four weighting schemes, the combined cue scheme performed the best for segmenting natural images. The linear scheme is only adequate for segmenting relatively simple synthetic images while the exponential and Gaussian scheme exhibit moderate performance.

In addition, I also investigated the suitability of using the $Ncut$ value as a measure for the image segmentation quality. We have seen some examples where the partition with the lowest $Ncut$ value is not as good a partition as compared to partition with higher $Ncut$ value. Although the $Ncut$ value is not the ultimate yardstick for the segmentation quality of an image, the $Ncut$ value still provides a good indication on the segmentation quality.

Image segmentation by the multilevel partition without refinement is able to give similar or better segmentation than segmenting the original image and offer shorter segmentation time. The multilevel scheme has less tendency of cutting through the objects in the image due to the coarsening of the graph. From the segmentation results of natural images, we see that sometimes partitioning the image by spectral method (without multilevel) still gives the best partition. For such as case, the multilevel partition with the inverse powering method can be used to obtain the same segmentation quality with shorter segmentation time.
6.2 Future Work

In this work, the multiclass spectral clustering and recursive bisection are used to partition the coarsest graph. A possible future research direction is to use other graph partitioning methods such as the spectral rounding and isoperimetric method for the base partition. With a faster base partitioning method, we can apply less coarsening to the original graph and still have the benefit of significant reduction in the segmentation time.

Another potential research direction is to perform the inverse iteration at each uncoarsening step. Instead of projecting the eigenvectors obtained at the coarsest level all the way back to $G_0$, the inverse iteration can be performed every time the graph is uncoarsened. As the graphs $G_i$ and $G_{i+1}$ are close to each other, the starting vector obtained from $G_{i+1}$ will also be close to the eigenvectors of the finer graph, $G_i$. Hence, the convergence should be fast and may even avoid the possibility of converging to an undesired higher or lower eigenvalue of $G_0$. 
Appendix A

Effects of Levels of Coarsening on the $Ncut$ value and Segmentation Time for Graph Bisection
Table A.1: $Ncut$ Values for various levels of coarsening (no refinement)

<table>
<thead>
<tr>
<th>Graph</th>
<th>Fiedler</th>
<th>%Δ in $Ncut$ for Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>airfoil1</td>
<td>0.0094</td>
<td>0.00</td>
</tr>
<tr>
<td>airfoil2</td>
<td>0.0135</td>
<td>4.44</td>
</tr>
<tr>
<td>airfoil3</td>
<td>0.0066</td>
<td>7.58</td>
</tr>
<tr>
<td>crack</td>
<td>0.0193</td>
<td>10.36</td>
</tr>
<tr>
<td>parc</td>
<td>0.0125</td>
<td>9.60</td>
</tr>
<tr>
<td>parcweb</td>
<td>0.0096</td>
<td>4.17</td>
</tr>
<tr>
<td>spiral</td>
<td>0.0044</td>
<td>0.00</td>
</tr>
<tr>
<td>tapir</td>
<td>0.0080</td>
<td>8.75</td>
</tr>
<tr>
<td>eppstein</td>
<td>0.0407</td>
<td>6.63</td>
</tr>
<tr>
<td>smallmesh</td>
<td>0.0679</td>
<td>3.39</td>
</tr>
</tbody>
</table>

Table A.2: Execution time for various levels of coarsening (no refinement)

<table>
<thead>
<tr>
<th>Graph</th>
<th>Fiedler</th>
<th>%Δ in execution time for Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>airfoil1</td>
<td>3.8173</td>
<td>-47.75</td>
</tr>
<tr>
<td>airfoil2</td>
<td>3.6234</td>
<td>-42.45</td>
</tr>
<tr>
<td>airfoil3</td>
<td>33.5522</td>
<td>-51.09</td>
</tr>
<tr>
<td>parc</td>
<td>1.4685</td>
<td>-40.18</td>
</tr>
<tr>
<td>parcweb</td>
<td>1.8494</td>
<td>-41.69</td>
</tr>
<tr>
<td>spiral</td>
<td>2.5510</td>
<td>-30.80</td>
</tr>
<tr>
<td>tapir</td>
<td>0.7462</td>
<td>-33.78</td>
</tr>
<tr>
<td>eppstein</td>
<td>0.2641</td>
<td>-45.10</td>
</tr>
<tr>
<td>smallmesh</td>
<td>0.0486</td>
<td>-5.76</td>
</tr>
</tbody>
</table>


Figure B-1: The process of coarsen the graph Airfoil1 from the original graph $G_0$ to the coarse graph $G_5$. (a) Original graph, $G_0$ (b)-(f) Coarse Graphs $G_1, G_2, ..., G_5$
Figure B-2: The process of coarsen the graph Airfoil2 from the original graph $G_0$ to the coarse graph $G_5$. (a) Original graph, $G_0$ (b)-(f) Coarse Graphs $G_1, G_2, ..., G_5$

Figure B-3: The process of coarsen the graph Airfoil3 from the original graph $G_0$ to the coarse graph $G_5$. (a) Original graph, $G_0$ (b)-(f) Coarse Graphs $G_1, G_2, ..., G_5$
Figure B-4: The process of coarsen the graph $\text{Parc}$ from the original graph $G_0$ to the coarse graph $G_5$. (a) Original graph, $G_0$ (b)-(f) Coarse Graphs $G_1, G_2, ..., G_5$

Figure B-5: The process of coarsen the graph $\text{Parcweb}$ from the original graph $G_0$ to the coarse graph $G_5$. (a) Original graph, $G_0$ (b)-(f) Coarse Graphs $G_1, G_2, ..., G_5$
Figure B-6: The process of coarsen the graph *Spiral* from the original graph $G_0$ to the coarse graph $G_5$. (a) Original graph, $G_0$ (b)-(f) Coarse Graphs $G_1, G_2, ..., G_5$

Figure B-7: The process of coarsen the graph *Eppstein* from the original graph $G_0$ to the coarse graph $G_5$. (a) Original graph, $G_0$ (b)-(f) Coarse Graphs $G_1, G_2, ..., G_5$
Figure B-8: The process of coarsen the graph Smallmesh from the original graph $G_0$ to the coarse graph $G_5$. (a) Original graph, $G_0$ (b)-(f) Coarse Graphs $G_1, G_2, ..., G_5$. 
Appendix C

Refinement Method Comparisons

Figure C-1: Segmentation of the graph crack by a) normalized cut, b) projection, c) greedy local refinement, and d) combination of inverse powering method with greedy local refinement
Figure C-2: Segmentation of the graph *airfoil2* by a) normalized cut, b) projection, c) greedy local refinement, and d) combination of inverse powering method with greedy local refinement.

Figure C-3: Segmentation of the graph *airfoil2* (Zoomed in)
Normalized Cut, $\text{Ncut} = 0.0066$

Figure C-4: Segmentation of the graph $\text{airfoil3}$ by a) normalized cut, b) projection, c) greedy local refinement, and d) combination of inverse powering method with greedy local refinement.

Figure C-5: Segmentation of the graph $\text{airfoil3}$ (Zoomed in)
Figure C-6: The process of coarsen the graph parc. (a) Original graph, $G_0$ (b)-(f) Coarse Graphs $G_1, G_2, \ldots, G_5$.

Figure C-7: Segmentation of the graph parcweb by a) normalized cut, b) projection, c) greedy local refinement, and d) combination of inverse powering method with greedy local refinement.
Figure C-8: Segmentation of the graph \textit{spiral} by a) normalized cut, b) projection, c) greedy local refinement, and d) combination of inverse powering method with greedy local refinement.

Figure C-9: Segmentation of the graph \textit{eppstein} by a) normalized cut, b) projection, c) greedy local refinement, and d) combination of inverse powering method with greedy local refinement.
Figure C-10: Segmentation of the graph *smallmesh* by a) normalized cut, b) projection, c) greedy local refinement, and d) combination of inverse powering method with greedy local refinement.
Appendix D

Effects of Levels of Coarsening on the $Ncut$ value and Segmentation Time for $k$-Way Graph Partition
Table D.1: $Ncut$ Values for various levels of coarsening ($k = 4$)

<table>
<thead>
<tr>
<th>Graph</th>
<th>Fiedler</th>
<th>$%\Delta$ in $Ncut$ for Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>airfoil1</td>
<td>0.0793</td>
<td>10.34</td>
</tr>
<tr>
<td>airfoil2</td>
<td>0.0926</td>
<td>10.80</td>
</tr>
<tr>
<td>airfoil3</td>
<td>0.0437</td>
<td>16.25</td>
</tr>
<tr>
<td>crack</td>
<td>0.1033</td>
<td>6.87</td>
</tr>
<tr>
<td>parc</td>
<td>0.0820</td>
<td>15.61</td>
</tr>
<tr>
<td>parcweb</td>
<td>0.0700</td>
<td>16.86</td>
</tr>
<tr>
<td>spiral</td>
<td>0.0447</td>
<td>6.94</td>
</tr>
<tr>
<td>tapir</td>
<td>0.0788</td>
<td>2.41</td>
</tr>
<tr>
<td>eppstein</td>
<td>0.2300</td>
<td>9.57</td>
</tr>
<tr>
<td>smallmesh</td>
<td>0.5793</td>
<td>17.04</td>
</tr>
</tbody>
</table>

Table D.2: $Ncut$ Values for various levels of coarsening ($k = 8$)

<table>
<thead>
<tr>
<th>Graph</th>
<th>Fiedler</th>
<th>$%\Delta$ in $Ncut$ for Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>airfoil1</td>
<td>0.2228</td>
<td>6.73</td>
</tr>
<tr>
<td>airfoil2</td>
<td>0.2409</td>
<td>7.89</td>
</tr>
<tr>
<td>airfoil3</td>
<td>0.1208</td>
<td>17.05</td>
</tr>
<tr>
<td>crack</td>
<td>0.3014</td>
<td>12.64</td>
</tr>
<tr>
<td>parc</td>
<td>0.2530</td>
<td>1.82</td>
</tr>
<tr>
<td>parcweb</td>
<td>0.2102</td>
<td>11.99</td>
</tr>
<tr>
<td>spiral</td>
<td>0.1727</td>
<td>8.28</td>
</tr>
<tr>
<td>tapir</td>
<td>0.4316</td>
<td>17.56</td>
</tr>
<tr>
<td>eppstein</td>
<td>0.8009</td>
<td>3.73</td>
</tr>
<tr>
<td>smallmesh</td>
<td>1.6852</td>
<td>16.66</td>
</tr>
</tbody>
</table>
Table D.3: Execution time for various levels of coarsening ($k = 4$)

<table>
<thead>
<tr>
<th>Graph</th>
<th>Fiedler</th>
<th>%Δ in execution time for Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>airfoil1</td>
<td>2.50</td>
<td>-40.01</td>
</tr>
<tr>
<td>airfoil2</td>
<td>2.55</td>
<td>-35.72</td>
</tr>
<tr>
<td>airfoil3</td>
<td>24.02</td>
<td>-47.76</td>
</tr>
<tr>
<td>crack</td>
<td>2.38</td>
<td>-21.95</td>
</tr>
<tr>
<td>parc</td>
<td>0.88</td>
<td>-39.55</td>
</tr>
<tr>
<td>spiral</td>
<td>1.48</td>
<td>-49.04</td>
</tr>
<tr>
<td>tapir</td>
<td>0.42</td>
<td>-26.27</td>
</tr>
<tr>
<td>eppstein</td>
<td>0.20</td>
<td>-39.18</td>
</tr>
<tr>
<td>smallmesh</td>
<td>0.05</td>
<td>-9.76</td>
</tr>
</tbody>
</table>

Table D.4: Execution time for various levels of coarsening ($k = 8$)

<table>
<thead>
<tr>
<th>Graph</th>
<th>Fiedler</th>
<th>%Δ in execution time for Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>airfoil1</td>
<td>1.83</td>
<td>-31.23</td>
</tr>
<tr>
<td>airfoil3</td>
<td>19.47</td>
<td>-42.07</td>
</tr>
<tr>
<td>parc</td>
<td>0.53</td>
<td>-30.50</td>
</tr>
<tr>
<td>parcweb</td>
<td>0.69</td>
<td>-24.14</td>
</tr>
<tr>
<td>spiral</td>
<td>0.66</td>
<td>-40.21</td>
</tr>
<tr>
<td>tapir</td>
<td>0.27</td>
<td>-12.03</td>
</tr>
<tr>
<td>eppstein</td>
<td>0.15</td>
<td>-27.54</td>
</tr>
<tr>
<td>smallmesh</td>
<td>0.04</td>
<td>-5.15</td>
</tr>
</tbody>
</table>
Appendix E

Comparison between the Recursive Bisection and Multiclass Spectral Clustering Method
Figure E-1: Comparison between the recursive bisection and multiclass spectral clustering method for 2, 4, and 8 partitions for graph crack.

Figure E-2: Comparison between the recursive bisection and multiclass spectral clustering method for 2, 4, and 8 partitions for graph parc.
Figure E-3: Comparison between the recursive bisection and multiclass spectral clustering method for 2, 4, and 8 partitions for graph parcweb.

Figure E-4: Comparison between the recursive bisection and multiclass spectral clustering method for 2, 4, and 8 partitions for graph spiral.
Figure E-5: Comparison between the recursive bisection and multiclass spectral clustering method for 2, 4, and 8 partitions for graph *eppstein*.

Figure E-6: Comparison between the recursive bisection and multiclass spectral clustering method for 2, 4, and 8 partitions for graph *smallmesh*.
Appendix F

Effects of Levels of Coarsening on the Quality of Image Segmentation
Figure F-1: Segmentation results for multilevel segmentation for 1 to 10 levels of coarsening without refinement. First Column: Landscape, Second column: Bird, Third column: Flower.
continued on the next page
Figure F-2: Segmentation results for multilevel segmentation for 1 to 10 levels of coarsening without refinement. First Column: Skater, Second column: Grass, Third column: MIT Dome.
continued on the next page
Figure F-3: Segmentation results for multilevel segmentation for 1 to 10 levels of coarsening without refinement. First Column: Panther, Second column: Butterfly, Third column: Taj Mahal.
Appendix G

Recursive Bisection Step

Figure G-1: Recursive Bisection
Figure G-2: Recursive Bisection for Heart
Appendix H

Natural Images Segmentation Data
Table H.1: Segmentation data for natural images segmentation

<table>
<thead>
<tr>
<th>Graph</th>
<th>k</th>
<th>Level</th>
<th>Ncut</th>
<th>Time (second)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Fiedler</td>
<td>multilevel</td>
</tr>
<tr>
<td>238011</td>
<td>4</td>
<td>2</td>
<td>0.0220</td>
<td>0.0094</td>
</tr>
<tr>
<td>163085</td>
<td>5</td>
<td>2</td>
<td>0.0112</td>
<td>0.0093</td>
</tr>
<tr>
<td>183055</td>
<td>4</td>
<td>3</td>
<td>0.0016</td>
<td>0.0032</td>
</tr>
<tr>
<td>130026</td>
<td>6</td>
<td>3</td>
<td>0.0457</td>
<td>0.0446</td>
</tr>
<tr>
<td>106020</td>
<td>8</td>
<td>4</td>
<td>0.0517</td>
<td>0.0629</td>
</tr>
<tr>
<td>100908</td>
<td>9</td>
<td>3</td>
<td>0.0421</td>
<td>0.0275</td>
</tr>
<tr>
<td>41033</td>
<td>4</td>
<td>1</td>
<td>0.0038</td>
<td>0.0044</td>
</tr>
<tr>
<td>42049</td>
<td>7</td>
<td>1</td>
<td>0.0067</td>
<td>0.0076</td>
</tr>
<tr>
<td>38092</td>
<td>6</td>
<td>1</td>
<td>0.0078</td>
<td>0.0081</td>
</tr>
<tr>
<td>86016</td>
<td>2</td>
<td>1</td>
<td>0.0003</td>
<td>0.0003</td>
</tr>
<tr>
<td>118020</td>
<td>7</td>
<td>2</td>
<td>0.0443</td>
<td>0.0554</td>
</tr>
<tr>
<td>385028</td>
<td>7</td>
<td>4</td>
<td>0.0130</td>
<td>0.0138</td>
</tr>
<tr>
<td>159091</td>
<td>6</td>
<td>4</td>
<td>0.0179</td>
<td>0.0255</td>
</tr>
<tr>
<td>188005</td>
<td>4</td>
<td>6</td>
<td>0.0080</td>
<td>0.0121</td>
</tr>
<tr>
<td>24063</td>
<td>2</td>
<td>3</td>
<td>0.0001</td>
<td>0.0004</td>
</tr>
<tr>
<td>293029</td>
<td>7</td>
<td>2</td>
<td>0.0225</td>
<td>0.0216</td>
</tr>
<tr>
<td>159029</td>
<td>3</td>
<td>2</td>
<td>0.0091</td>
<td>0.0101</td>
</tr>
<tr>
<td>42044</td>
<td>5</td>
<td>1</td>
<td>0.0548</td>
<td>0.0461</td>
</tr>
</tbody>
</table>

Continued on the next page
Continued from the previous page

<table>
<thead>
<tr>
<th>Graph</th>
<th>k</th>
<th>Level</th>
<th>Ncut</th>
<th>Time (second)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Fiedler</td>
<td>multilevel</td>
</tr>
<tr>
<td>76002</td>
<td>3</td>
<td>2</td>
<td>0.0030</td>
<td>0.0032</td>
</tr>
<tr>
<td>100075</td>
<td>5</td>
<td>3</td>
<td>0.0242</td>
<td>0.0215</td>
</tr>
<tr>
<td>103041</td>
<td>6</td>
<td>3</td>
<td>0.0259</td>
<td>0.0408</td>
</tr>
<tr>
<td>295087</td>
<td>8</td>
<td>3</td>
<td>0.0394</td>
<td>0.0401</td>
</tr>
<tr>
<td>103070</td>
<td>8</td>
<td>4</td>
<td>0.0790</td>
<td>0.0818</td>
</tr>
<tr>
<td>105025</td>
<td>5</td>
<td>6</td>
<td>0.0085</td>
<td>0.0154</td>
</tr>
<tr>
<td>123074</td>
<td>10</td>
<td>3</td>
<td>0.0631</td>
<td>0.0573</td>
</tr>
<tr>
<td>260058</td>
<td>2</td>
<td>1</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>220075</td>
<td>4</td>
<td>7</td>
<td>0.0038</td>
<td>0.0064</td>
</tr>
<tr>
<td>361084</td>
<td>7</td>
<td>3</td>
<td>0.0443</td>
<td>0.0427</td>
</tr>
<tr>
<td>189003</td>
<td>8</td>
<td>2</td>
<td>0.0035</td>
<td>0.0204</td>
</tr>
<tr>
<td>296059</td>
<td>4</td>
<td>1</td>
<td>0.0087</td>
<td>0.0089</td>
</tr>
<tr>
<td>253036</td>
<td>2</td>
<td>3</td>
<td>0.0003</td>
<td>0.0004</td>
</tr>
<tr>
<td>176039</td>
<td>3</td>
<td>3</td>
<td>0.0055</td>
<td>0.0068</td>
</tr>
<tr>
<td>170054</td>
<td>5</td>
<td>6</td>
<td>0.0069</td>
<td>0.0127</td>
</tr>
<tr>
<td>55067</td>
<td>4</td>
<td>1</td>
<td>0.0014</td>
<td>0.0017</td>
</tr>
<tr>
<td>163062</td>
<td>10</td>
<td>2</td>
<td>0.0573</td>
<td>0.0623</td>
</tr>
<tr>
<td>140075</td>
<td>7</td>
<td>4</td>
<td>0.0096</td>
<td>0.0189</td>
</tr>
<tr>
<td>86000</td>
<td>3</td>
<td>5</td>
<td>0.0055</td>
<td>0.0028</td>
</tr>
<tr>
<td>198054</td>
<td>7</td>
<td>1</td>
<td>0.0522</td>
<td>0.0454</td>
</tr>
<tr>
<td>187071</td>
<td>3</td>
<td>2</td>
<td>0.0096</td>
<td>0.0041</td>
</tr>
<tr>
<td>23025</td>
<td>4</td>
<td>1</td>
<td>0.0050</td>
<td>0.0052</td>
</tr>
<tr>
<td>176019</td>
<td>5</td>
<td>3</td>
<td>0.0496</td>
<td>0.0315</td>
</tr>
<tr>
<td>89072</td>
<td>9</td>
<td>7</td>
<td>0.0139</td>
<td>0.0884</td>
</tr>
<tr>
<td>78004</td>
<td>4</td>
<td>3</td>
<td>0.0152</td>
<td>0.0145</td>
</tr>
<tr>
<td>167083</td>
<td>5</td>
<td>2</td>
<td>0.0034</td>
<td>0.0044</td>
</tr>
<tr>
<td>268002</td>
<td>7</td>
<td>7</td>
<td>0.0216</td>
<td>0.0511</td>
</tr>
<tr>
<td>22093</td>
<td>8</td>
<td>1</td>
<td>0.0373</td>
<td>0.0419</td>
</tr>
<tr>
<td>285079</td>
<td>5</td>
<td>10</td>
<td>0.0333</td>
<td>0.0416</td>
</tr>
<tr>
<td>157036</td>
<td>7</td>
<td>1</td>
<td>0.0188</td>
<td>0.0211</td>
</tr>
</tbody>
</table>

143
Bibliography


