Proposed Abstract

In 2008, Reid Anderson and Kevin Lang published a paper describing a new algorithm which improves a given graph partition in deterministic polynomial time. Their algorithm solves a series of s-t minimum cut problems on a modified version of the graph to produce a new partition which is at least as sparse as any subset of the old partition. This semester I hope to extend their framework to create an algorithm to improve multi-way sparse partitions of graphs.

Introduction to Sparse Graph Partitioning

Finding the sparsest cut in an arbitrary graph G is a classic problem of theoretical computer science. Given a graph $G=(V,E)$, let $\pi(v): V \rightarrow \mathbb{R}^+$ be a function assigning a positive weight to each vertex, and let edge $(u,v) \in E$ be weighted by a function $w(u,v): E \rightarrow \mathbb{R}$. For any $S \subseteq V$, define the edge border of that set, $\delta(S)$, to be the sum of the weights of the edges with one endpoint in S and the other endpoint in $V \setminus S$. Also, for $S \subseteq V$ define $\pi(S)$ to be the sum of the weights of vertices in S, $\pi(S) = \sum_{v \in S} \pi(v)$. The sparse cut problem can then be described as follows: Given $G=(V,E)$, find $S \subseteq V$ minimizing $Q(S) = \frac{\delta(S)}{\min(\pi(S), \pi(S))}$ where $\overline{S}$ denotes $V \setminus S$. Here $Q(S)$ denotes the “quotient score” of S.

Finding sparse graph cuts has numerous applications in computer science, especially in distributed computing. Suppose you wish to perform a calculation on a large graph using a multicomputer; you then need to assign each node of the computer a segment of the graph to work on. Using a sparse graph cut to distribute the work will minimize the amount of inter-node communication necessary to perform the computation, thus optimizing performance.

Unfortunately the sparse cut problem, as stated above, is NP-complete, so we do not have an efficient algorithm to solve this problem. However, computer scientists have devised numerous approximation algorithms to the sparsest cut problem, the best of which give you an $O(\log^{1/2} n)$ approximation of the sparsest cut in the graph.

The sparse graph cut problem can also be generalized to consider multi-way cuts in graphs. Suppose we create a k-way partition $P$ of a graph $G=(V,E)$, that is a collection of k disjoint subsets of V, $S_1, S_2, \ldots, S_k$, such that they cover all of V. ($\forall i, j, S_i \cap S_j = \emptyset$ and $\bigcup S_i = V$). The multi-way cut problem is to find a k-way partition minimizing an objective function which describes the quality of the cut. Different authors use different objective functions, but a commonly used objective function is the sum of the quotient scores of the
individual cuts, e.g. $Q(P) = \sum_i Q(S_i)$. The k-way sparse cut problem under this objective function is also NP-Complete.

**Goals of Project**

The goal of my project is to extend an algorithm developed by Reid Anderson and Kevin Lang so that it can handle multi-way cuts in graphs. Their algorithm, entitled “Improve”, takes a proposed 2-way cut of a graph $G$, defined by the set $A$, and improves the quality of the cut in deterministic polynomial time, outputting a cut $S$ with improved cut quality. Specifically, $S$ is guaranteed to have a cut quality at least as good as the best subset of $A$. ($\forall C \subseteq A, Q(S) \leq Q(C)$)

Furthermore, if $C$ is a subset of $V$ satisfying $\frac{\pi(A \cap C)}{\pi(C)} \geq \frac{\pi(A)}{\pi(V)} + \varepsilon \frac{\pi(A)}{\pi(V)}$ for some $\varepsilon > 0$ (in other words, if $C$ has more in common with $A$ than one would expect from a random subset of $V$), then we are guaranteed that $Q(S) \leq \frac{1}{\varepsilon} Q(C)$.

The algorithm works by finding a cut of $G$ which minimizes a modified quotient score $\tilde{Q}_a(S)$ of $S$ with respect to $A$, which is defined as $\tilde{Q}_a(S) = \frac{\partial(S)}{D_a(S)}$ where $D_a(S) = \pi(S \cap A) - \pi(S \cap A) \frac{\pi(A)}{\pi(A)}$. To account for the fact that $D_a$ can be $\leq 0$, we define $\tilde{Q}_a(S)$ such that if $D_a(S) \leq 0$, then $\tilde{Q}_a(S) = \infty$. From henceforth we will rename $\frac{\pi(A)}{\pi(A)}$ as $f(A)$. Note that for any set $S$, $\tilde{Q}_a(S) \geq Q(S)$, so minimizing the modified quotient cut will not necessarily minimize the actual quotient score. However, it can be shown that by producing a cut that minimizes the modified quotient score, we will produce a set which has the properties described above.

The algorithm finds a cut minimizing $\tilde{Q}_a(S)$ by solving a series of minimum s-t cut problems. In each s-t cut problem, parameterized by $\alpha \in [0, \infty)$, it adds vertices $s$ and $t$ to $G$ to create a new graph $G_{A,\alpha}$. It then adds an edge from $s$ to each vertex $v \in A$ with weight $\alpha \pi(v)$, and adds an edge from all $u \in \overline{A}$ to $t$ with weight $\alpha \pi(u) f(A)$. S-T cuts in this new graph $G_{A,\alpha}$ are equated with a cut $S$ in $G$ by lumping $s$ with $S$ and $t$ with $\overline{S}$.

The nice property of this setup that if there exists a subset $X \subset V$ with $\tilde{Q}_a(X) < \alpha$, then the minimum s-t cut in this new graph $G_{A,\alpha}$, $S_{\alpha}$, satisfies $\tilde{Q}_a(S_{\alpha}) < \alpha$. Thus the Improve algorithm works as follows: It initializes $\alpha_1$ with value $\tilde{Q}(A)$. For each step of the algorithm, it solves the s-t cut problem on $G_{A,a_i}$. The resulting min s-t cut $S_{\alpha_i}$ is guaranteed to have $\tilde{Q}_a(S_{\alpha_i}) < \alpha_i$. We then set $\alpha_{i+1} = \tilde{Q}_a(S_{\alpha_i})$ and repeat until our $\alpha$ no longer decreases (so our cut cannot be improved). At this point we output the resulting set. By the above observation, the
quality of our cut (as measured by $\tilde{Q}$) will improve each step until we find a cut with minimal $\tilde{Q}$.

Anderson and Lang have showed that this algorithm will conduct at most $\pi(V)^2$ iterations on a graph with integer weights and at most $m = |E|$ iterations on an unweighted graph.

My goal this semester is to extend the framework of this algorithm to improve multi-way graph partitions. Since Anderson and Lang’s algorithm is based on network flow, I plan to investigate ways of using multi-commodity flow to help improve multi-way cuts. However, multi-commodity flow is NP-complete, even so the flow solutions used in the algorithm will have to be approximate. This will introduce another source of approximation into my algorithm. Furthermore, I may have to redefine the metric used to measure the sparseness of a multi-way cut in order to facilitate the proof of the correctness of my algorithm.

**Deliverables**

Depending on the progress I make, I hope to produce an algorithm, modeled after the algorithm devised by Anderson and Lang, which improves multi-way cuts in graphs. I also plan to produce a proof of the correctness of the algorithm as well as time bounds on the performance of the algorithm. During the course of the semester, I may find it necessary to code my algorithm to test its runtime; if I happen to do this I will turn this in as well. Time permitting, I may also write code to empirically test my algorithm.

At the end of the semester, I will produce a final report and website detailing my results, complete with my proof of the correctness and runtime of the algorithm. Furthermore, I will deliver a 30-60 minute presentation on my research to the Mathematics Department.