# 1 Lecture 13, January 13, 2000

We present two randomized algorithms.

## 1.1 Minimum cuts

Let $G(V, E)$ be a graph. A $k$-cut is a partition of $V$ into $k$ disjoint parts $V_1, \ldots, V_k$. The size of the cut is the number of edges cut, namely $\sum_{i<j} E(V_i, V_j)$. We wish to find $k$-cuts of minimum size, for constant values of $k$.

The case $k = 2$ is known as the minimum cut problem. A standard approach for solving it uses minimum $s, t$-cuts, where an $s, t$-cut is a 2-cut for which $s \in V_1$ and $t \in V_2$. Minimum $s, t$-cuts can be found in polynomial time using max-flow algorithms, using the max-flow min-cut duality. Then, fixing some vertex as $s$ and letting $t$ range over all other vertices, one solves $n-1$ $s, t$-cut problems and takes the minimum of these solutions as the minimum cut.

This approach does not generalize to $k \geq 3$, because finding a cut that separates three given vertices is NP-hard. Nevertheless, minimum $k$-cut can be solved in polynomial time when $k$ is constant. The problem becomes NP-hard when $k$ changes as part of the input.

We present a simple randomized algorithm for finding minimum $k$-cuts. We start with a version that is applicable for $k = 2$, and then slightly modify it so as to apply to larger values of $k$. The algorithm is applicable also to multigraphs that have parallel edges. It uses the *edge contraction* operation, where an edge $e = (u, v)$ is contracted by merging the two vertices $u$ and $v$ into one super-vertex $\{u, v\}$, and keeping all edges that connect $u$ and $v$ to the rest of the graph (with their appropriate multiplicity). Self loops are eliminated. We assume without loss of generality that the graph is connected, as otherwise a minimum cut contains no edges. The basic algorithm is very simple:

- *Repeatedly select an edge at random and contract it, until only two super-vertices remain. Make the set of vertices that corresponds to one of them $V_1$, and the set of vertices that correspond to the other $V_2$.*

The graph $G$ may have several minimum cuts (e.g., for a clique, every individual vertex is a minimum cut). For sake of analysis, fix one particular minimum cut $S, V \setminus S$, with cut edges $E[S, V \setminus S]$. The basic algorithm outputs this cut iff at every iteration, the contracted edge did not belong to $E[S, V \setminus S]$. Let $M = |E[S, V \setminus S]|$ and note that every cut contains at least $M$ edges.

Let $G_n = G, G_{n-1}, \ldots, G_2$ be the sequence of graphs produced by the basic algorithm. In every such graph, every super-vertex has degree at least $M$, because otherwise there is a cut with less that $M$ edges. Hence for every $3 \leq j \leq n$, the graph $G_j$ contains at least $jM/2$ edges, and the probability that the random edge is from $E[S, V \setminus S]$ is at most $M/(jM/2) = 2/j$. Hence the basic algorithm suceeds with probability

$$\Pi_{j=n}^{3}(1 - \frac{2}{j}) = \Pi_{j=n}^{3} \frac{j-2}{j} = \frac{n(n-1)}{2}.$$

Hence if we repeat the basic algorithm $cn^2$ times, we have probability at least $1 - e^{2c}$ of finding the minimum cut $S, V \setminus S$. Moreover, every minimum cut has probability at least

$2/n(n-1)$ of being output by the basic algorithm. As probabilities sum up to at most 1, we have the following corollary.

**Corollary 1** *A connected graph has at most $\binom{n}{2}$ minimum cuts.*

To find a minimum $k$-cut for $k \geq 3$, we can employ the same basic algorithm until $k$ super-vertices remain. However, to simplify the analysis, let us assume that we employ the basic algorithm only until $2k$ super-vertices remain, and thereafter find the best $k$-cut in $G_{2k}$ via exhaustive search. The number of possibilities to try in this last step is only a function of $k$ (less than $k^{2k}$) and as $k$ is assumed to be constant, the exhaustive search part does not significantly effect the running time.

Let $M$ be the number of edges cut by an optimal $k$-cut in $G$. Then for every $2k+1 \leq j \leq n$, the average degree in $G_j$ is at least $M/(k-1)$, because otherwise taking the $k-1$ super-vertices of lowest degree gives a $k$-cut with less than $M$ edges. Hence the total number of edges in $G_j$ is at least $jM/2(k-1)$ and the probability of randomly contracting a cut edge is at most $2(k-1)/j$. Hence the basic algorithm suceeds with probability

$$\Pi_{j=n}^{2k+1}(1-\frac{2(k-1)}{j}) = \Pi_{j=n}^{2k+1}\frac{j-2(k-1)}{j} \leq n^{2(k-1)}.$$

Repeating the basic algorithm $O(n^{2(k-1)})$ times we have high probability of producing a minimum $k$-cut.

## 1.2 Domatic number

A set $S \subset V$ of vertices in a graph $G(V,E)$ is a *dominating set* if every vertex in $V \setminus S$ is adjacent to some vertex in $S$. Finding a minimum dominating set in a graph is NP-hard. In fact, this problem is equivalent to set cover via the following reductions. An instance of dominating set can be reduced to an instance of set cover by making each vertex an element and each set that includes a vertex and all its neighbors a set. An instance of set cover can be reduced to an instance of dominating set by making each element a vertex, each set a vertex, connecting each set to all its elements, and making a clique of all sets. Both these reductions exactly preserve the cardinality of the minimum solution. By this equivalence with set cover, it follows that minimum dominating set is hard to approximate within a ratio of nearly $\ln n$. (Here note a subtle point, that in the reduction from set cover to dominating set, the parameter $n$ changes. For set cover it is the number of elements. For dominating set it is the number of vertices. The number of vertices is the sum of number of elements $N$ and number of sets $M$. Hence the $\ln n$ hardness result for dominating set assumes that $\ln N \simeq \ln(N+M)$ for instances of set cover that are hard to approximate within nearly $\ln N$, which is indeed true.) It also follows that the natural greedy algorithm achieves the best possible approximation ratio for dominating set (as a function of $n$, up to low order terms).

A *domatic partition* of $G$ is a partition of its vertices into disjoint dominating sets. The *domatic number* is the maximum number of dominating sets in a domatic partition. It can be shown that approximating domatic number within a factor of nearly $1/\ln n$ is hard (unless NP has $n^{O(\log\log n)}$-time algorithms). By its close connection to dominating set, one may hope that domatic number can be approximated within a $1/\ln n$ factor by a greedy algorithm. This is indeed true, but the greedy algorithm achieving this approximation ratio is not as natural. For example, the approach of iteratively picking dominating sets, where each dominating set is chosen greedily and disjoint from the previous ones, gives an approximation ratio of only roughly $1/\sqrt{n}$.

Rather than guess a greedy algorithm for domatic number, we shall develop one in a semi-systematic way. The approach is as follows:

1. Design a randomized algorithm for the problem.
2. Derandomize the algorithm using the method of conditional probabilities.
3. The derandomized algorithm has an interpretation as a greedy algorithm.

In this lecture we will develop the randomized algorithm. In the next lecture we will see how to derandomize it.

For simplicity of notation, let the set of neighbors of a vertex include the vertex itself. Let $\delta$ be the minimum cardinality of a neighborhood of a vertex in $G$. Then the domatic number can be at most $\delta$, because every dominating set contains at least one vertex from this neighborhood. We shall show how to find a domatic partition with roughly $\delta/\ln n$ dominating sets, which is a $1/\ln n$ approximation.

Let $l = \delta/\ln(n \ln n)$. The randomized algorithm works as follows:

1. Give each vertex a color between 1 and $l$ at random.
2. The resulting domatic partition consists of those color classes that are dominating sets, with vertices in other color classes added to an arbitrary dominating set.

To analyse the above algorithm, let $A_{v,c}$ denote the event that the neighborhood of vertex $v$ does not include color $c$. Then $Pr[A_{v,c}] \leq (1 - 1/l)^\delta \leq e^{-\ln(n \ln n)} = 1/n \ln n$. Treating the $A_{v,c}$ as 0/1 indicator variables, let $A = \sum_{v,c} A_{v,c}$. Using linearity of expectation, the expectation of $A$ satisfies

$$E[A] = \sum_{v,c} E[A_{v,c}] = \sum_{v,c} Pr[A_{v,c}] \leq \sum_{v,c} \frac{1}{n \ln n} = \frac{l}{\ln n}.$$

It follows that for some choice of colors to the vertices, the number of bad events is at most $l/\ln n$, and this then also bounds the number of color classes involved in bad events. Hence at least $l(1 - 1/\ln n)$ color classes are dominating sets. Plugging in the value of $l = \delta/\ln(n \ln n)$, we find that the domatic number is at least $(1-\epsilon)\delta/\ln n$, where $\epsilon < (1 + \ln \ln n)/\ln n$. Using the fact that $A$ has probability at most 1/2 of exceeding its expectation by a factor of two, we get a randomized algorithm that with high probability finds a domatic partition with $(1-\epsilon)\delta/\ln n$ dominating sets, for a somewhat larger value of $\epsilon$. In the next lecture we will see how to obtain a deterministic algorithm.

**References:**


- D. Karger and C. Stein. "A new approach to the minimum cut problem". *Journal of the ACM*, 43(4), 1996.
- U. Feige, M. Halldorsson, G. Kortsarz. "Approximating the domatic number". To appear in STOC 2000.