Bisection Sketch

Zhifeng Sun

October 31, 2008

In 2002, Feige and Krauthgamer published a combinatorial algorithm of bisection problem with $O(\log^2 n)$-approximation ratio. This document is trying to give a high level structure of the proof and present some intuition behind it.

1 Upper bound explanation

In the decomposition step, as shown in Figure 1, assume that (C, F) is any bisection, and (U_1, U_2) is the decomposition by amortized cut. The cost of bisection can be calculated as follows.

$$\mathrm{Cut}\,(C,F) = \mathrm{Cut}\,(C_1,F_1) + \mathrm{Cut}\,(C_2,F_2) + \mathrm{e}\,(C_1,F_2) + \mathrm{e}\,(C_2,F_1)$$

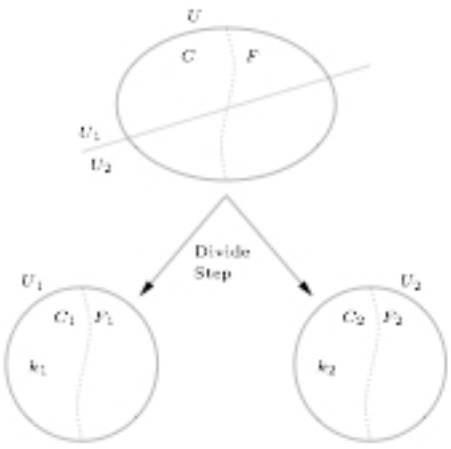

Figure 1: Divide and conquer

We can upper bound the cost by charging more edges, as follows.

$$\mathrm{Cut}\,(C,F) \le \mathrm{Cut}\,(C_1,F_1) + \mathrm{Cut}\,(C_2,F_2) + \mathrm{e}\,(C_1,U_2) + \mathrm{e}\,(C_2,U_1)$$

or

$$\mathrm{Cut}\,(C, F) \leq \mathrm{Cut}\,(C_1, F_1) + \mathrm{Cut}\,(C_2, F_2) + \mathrm{e}\,(U_1, F_2) + \mathrm{e}\,(U_2, F_1)$$

In each divide step, no matter which way we charge, it is clearly an upper bound. And the edges we charged in this step will never be charged again in the following divide steps. So we can mix these two kinds of charges in divide steps, and pick the one which gives a better bound.

2 What is labeling

Labeling is the technique that helps us decide which kind of charge to use in each step, i.e. do we charge on F_i or C_i. The crucial theorem they proved in the paper is, in each decomposition step if you always charge the "small" part (i.e. if you charge C_i then $|C_i| \leq \alpha |C_i \cup F_i|$, for $\frac{1}{2} \leq \alpha \leq 1$), then this upper bound is within $O(\log^2 n)$ times the real bisection cost.

So they create a family of labelings based on the decomposition tree (notice this decomposition tree is unique), in which it is guaranteed to have a labeling that always charges the "small" part with respect to the optimal bisection. If for all the bisection and labeling pairs we can compute the minimum one, then we can approximate minimal bisection problem within $O(\log^2 n)$. However the number of labelings in that family is exponential. We cannot afford to compute one by one and pick the minimum. That is where dynamic programming comes into place.

3 Dynamic programming

This section points out one crucial point for their dynamic programming. Notice the depth of the decomposition tree may be $O(n)$, but the marked (i.e. labeled) nodes on the path, from root to each leaf, is $O(\log n)$. And this makes the dynamic programming in polynomial time, because each step they need to try all the possible color combinations of the marked nodes along the path from the corresponding node to the root.