Randomness and Computation
Joint Workshop "New Horizons in Computing" and "Statistical Mechanical Approach to Probabilistic Information Processing" (18-21 July, 2005, Sendai, Japan)

Bisection Problem on Bipartite Graphs

Mikael Onsjö [1]
Dept. of Computer Engineering
Chalmers Univ. of Technology, Sweden

Osamu Watanabe [2]
Dept. of Math. and Comput. Sci
Tokyo Inst. of Technology, Japan

1 Problem Description

Bisection Problem is to patition a given graph G into subgraphs G_1 and G_2 of equal size with minimum number of cut edges, i.e., edges between G_1 and G_2. This problem has been studied extensively by many researchers [1, 2, 4], and several algorithms have been proposed and analyzed both mathematically and experimentally. Here we propose yet another algorithm for this problem.

In this paper, we consider only bipartite graphs $G = (V_1, V_2, E)$ such that $|V_1| = |V_2| = n$ for some size parameter n. That is, E is a subset of $\{(i, j) : i \in V_1, j \in V_2\}$. We will use i to denote vertices in V_1 and j to denote vertices in V_2. An *equal size partition* of $V_1 \cup V_2$ is a partition (C^+, C^-) such that $V_1 = C^- \cap V_1 = C^- \cap V_1 = C^+ \cap V_2 = C^- \cap V_2 = n/2$. (We assume that n is even.)

The Bisection Problem for general graphs is NP-hard, and it is easy to modify the proof to show the NP-hardness of our problem. Thus, one may not be able to hope for an efficient algorithm solving the problem for all instances. On the other hand, the problem seems not so hard *on average* under some reasonable probability model on input graphs. In fact, under "planted models", which we explain below, several algorithms have been proposed and some of them are shown to solve the problem [1, 2, 3, 4] in polynomial-time on average.

Let us define a *planted model*, a probability model for input graph distribution. Consider any even n, and let V_1 and V_2 be respectively a set of n vertices, and let (C^+, C^-) be its equal size partition. For given parameters p and r, we consider the following random generation of a bipartite graph (V_1, V_2, E): For any vertex $i \in V_1$ and $j \in V_2$, add an edge (i, j) to E, with probability p if both i and j are in C^+ (or, respectively, in C^-) and with probability r otherwise. For any size parameter n, and parameters p and r, this model defines a probability distribution on bipartite of size $2n$. We discuss the average performace of algorithms under this probability distribution. It has been shown [1] that if $p - r$ is large engough (more specifically, $p - r = \Omega(n^{-2})$, then a *planted solution*, the partition used to generate an instance to the problem, is a unique solution for the generated instance with high probability. Thus, such parameters p and r, the goal of bisection algorithms is to find a planted solution.

There is another natural way to formalize a problem for panted models. For any graph $G = (V_1, V_2, E)$ and its partition (C^+, C^-), and for given parameters p and r, the following is the probability that G is generated from (C^+, C^-) as above.

$$\Pr[\, G \text{ is generated from } (C^+, C^-)\,] \;=\; \prod_{(i,j)\in E} p^{\delta(i,j)} r^{1-\delta(i,j)} \cdot \prod_{(i,j)\in \overline{E}} (1-p)^{\delta(i,j)} (1-r)^{1-\delta(i,j)},$$

where $\overline{E} = \{(i, j) : i \in V_1, j \in V_2, (i, j) \notin E\}$, and δ is defined so that, for any i, j, $\delta(i, j) = 1$ if both i and j are in C^+ (or in C^-), and $\delta(i, j) = 0$ otherwise. We call this probability the *likelihood* of (C^+, C^-). Now *Most Likely Partition Problem* (for bipartite graphs) is, for given bipartite graph $G = (V_1, V_2, E)$, and parameters p and r, $0 < r < p < 1$, to find a partition (C^+, C^-) with the highest likelihood for G w.r.t. the parameters p and r.

Note that if $p - r$ is large enough, it can be shown again that that a planted solution is with high probability the solution of the Most Likely Partition Problem. In other words, under planted models and with reasonable choice of p and r, these two problems ask for the same answer. Solutions may differ, however, if $p - r$ is not so large.

program CompBelief((V_1, V_2, E), p, r);
% This algorithm computes "classification belief" for
% each vertices in $V_1 \cup V_2$.
begin
  set all x_i and y_j to 0;
  repeat MAXSTEP times **do** {
    $x_1 \leftarrow +\infty$;
    for each $j \in V_2$,
    $y_j \leftarrow \sum_{i \in N(j)} h_+ \cdot \mathrm{Th}_+(x_i) - \sum_{i \notin N(j)} h_- \cdot \mathrm{Th}_-(x_i)$;
    for each $i \in V_1$,
    $x_i \leftarrow \sum_{j \in N(i)} h_+ \cdot \mathrm{Th}_+(y_j) - \sum_{j \notin N(i)} h_- \cdot \mathrm{Th}_-(y_j)$;
    if all beliefs exceed the threshold **then break**;
  }
  output classification $(\mathrm{sg}(x_1), ..., \mathrm{sg}(x_n), \mathrm{sg}(y_1), ..., \mathrm{sg}(y_n))$;
end-program

parameters & functions

$$a_- = \frac{1-p}{1-r}, \quad a_+ = \frac{p}{r},$$
$$h_- = \left|\frac{a_- - 1}{a_- + 1}\right|, \quad h_+ = \left|\frac{a_+ - 1}{a_+ + 1}\right|,$$
$$\mathrm{th}_- = \left|\frac{\ln a_-}{h_-}\right|, \quad \mathrm{th}_+ = \left|\frac{\ln a_+}{h_+}\right|,$$
$$\mathrm{Th}_+(z) = \mathrm{sg}(z) \cdot \min(z, \mathrm{th}_+),$$
$$\mathrm{Th}_-(z) = \mathrm{sg}(z) \cdot \min(z, \mathrm{th}_-), \quad \text{and}$$
$$\mathrm{sg}(z) = \begin{cases} +1, & \text{if } z > 0, \\ 0, & \text{if } z = 0, \text{ and} \\ -1, & \text{otherwise} \end{cases}$$

Figure 1: Algorithm for the Bisection Problem

2 Our Algorithm and Result

We propose an algorithm (Figure 1) that is obtained by a slight modification of Pearl's belief propagation (BP in short) [6]. Belief propagation is a way of inferring probabilities such as likelihood. It is based on message passing through a Bayesian network. Although this method does not work in general if some loop exists in the network, it has been shown that even with a graph having loops, the BP works to some extent *on average*, and the BP has been used successfully for various applications, see, e.g., [5]. For our modification, we simply removed one restriction from belief propagation's rule for updating the belief on each vertex's classification in a Bayesian network, which yields a slightly different threshold linear system. The new algorithm is still quadratic but conceptually simpler. From our experiments, the new algorithm almost always converges whereas the original belief propagation sometimes contained limit cycles.

We show that this algorithm in fact yields a planted solution after the first round if n is sufficiently large for r and $p - r$. More specifically, we proved that for some constant ϵ_0 and any p and r, we have

$$\Pr[\text{ the algorithm gives the planted solution for } V_1 \text{ after the first round }] \ \geq \ 1 - 2n \cdot e^{-\epsilon_0 \frac{n}{r(p-r)^4}}.$$

A solution for V_2 is obtained in the next round.

References

[1] R.B. Boppana, Eigenvalues and graph bisection: an average-case analysis, in *Proc. Symposium on Foundations of Computer Science*, 280-285, 1987.

[2] T. Bui, S. Chaudhuri, F. Leighton, and M. Spiser, Graph bisection algorithms with good average behaviour, in *Combinatorica*, 7, 171–191, 1987.

[3] D. Dubhashi, L. Laura, and A. Panconesi, Analysis and experimental evaluation of a simple algorithm for oollaborative filtering in planted partition models, in *Proc. 23rd Conf. on Foundations of Software Tech. and Theoret. Comp. Sci.*, Lecture Notes in CS 2914, 168–182, 2003.

[4] M. Jerrum and G. Sorkin, The Metropolis algorithm for graph bisection, *Discrete Appl. Math*, 82(1-3), 155–175, 1998.

[5] R. McEliece, D. MacKay, and J. Cheng, Turbo decoding as an instance of Pearl's "Belief Propagation" algorithm, in *IEEE J. on Selected Areas in Comm.*, 16(2), 1998.

[6] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers Inc., 1988.