ISSN 1342-2812

Research Reports on Mathematical and Computing Sciences

Simple Algorithms for Graph Partition Problems

Mikael Onsjö and Osamu Watanabe

Aug. 2005, C–212

Department of
Mathematical and
Computing Sciences
Tokyo Institute of Technology

SERIES C: Computer Science

Simple Algorithms for Graph Partition Problems

Mikael Onsjö*
Dept. of Computer Engineering, Chalmers Univ. of Technology, Sweden
(mikael@odinalake.net)
Osamu Watanabe†
Dept. of Math. and Comp. Sci., Tokyo Inst. of Technology, Japan
(watanabe@is.titech.ac.jp)

Research Report C-212

Abstract

Based on the Belief Propagation Method, we propose simple and deterministic algorithms for some NP-hard graph partitioning problems, such as the Most Likely Partition problem and the Graph Bisection problem. These algorithms run in $\mathcal{O}(n+m)$ or $\mathcal{O}((n+m)\log n)$ time on graphs with n vertices and m edges. For their average case analysis, we consider the planted solution model and prove that they yield an correct answer with high probability for large range of probabilistic parameters.

1 Introduction

We propose simple algorithms for some NP-hard graph partition problems including the Graph Bisection problem. The algorithms are *deterministic* and run in linear time, i.e., $\mathcal{O}(n+m)$ time w.r.t. vertex size n and edge size m (for some problems, $\mathcal{O}((n+m)\log n)$ time), and they solve their target problems with high probability for randomly generated instances.

We first specify our main problem — Most Likely Partition problem. This problem is, for a given graph (and probability parameters), to find a partition that is most likely to generate this graph under a certain random model explained below.

For defining this problem, we need some notions and notations. In this paper we consider simple undirected graphs. A graph is given by a pair $G = (V, E)$ of a vertex set V and an edge set E. For simplicity, we identify V as $\{1, ..., |V|\}$, and assume that E is a set of pairs (i, j) such that $i < j$. By $V \times V$, we denote the set $\{(i, j) : i, j \in V \text{ and } i < j\}$. By $\overline{E}$ we denote a set $V \times V - E$. A *partition* of a set V is a pair V_1 and V_2 of disjoint sets such that $V_1 \cup V_2 = V$, and it is called an *equal size partition* if $|V_1| = |V_2|$. For a given partition V_1 and V_2 of a vertex set V, its cut edge is an edge between $u \in V_1$ and $v \in V_2$.

---

*This work has been done while visiting Tokyo Inst. of Technology.

†Supported in part by a Grant-in-Aid for Scientific Research on Priority Areas "Statical-Mechanical Approach to Probabilistic Information Processing" 2002-2005.

Our random model specifies a way to generate a graph from its partition. For any n, consider a set of vertices V of size[1] $2n$, and choose some of its elements uniformly at random, generating a partition V_+ and V_-. Define a function $c : V \to \{+1, -1\}$ so that $c(i) = +1$ if $i \in V_+$ and $c(i) = -1$ if $i \in V_-$; this function c is called a *characteristic function* w.r.t. the partition (V_+, V_-). For given parameters p and r, we consider the following random generation of a graph $G = (V, E)$: for any vertices $i, j \in V$, put an edge (i, j) to E with probability p if $c(i) = c(j)$, and put an edge (i, j) to E with probability r if $c(i) \neq c(j)$. For any size parameter n, and parameters p and r, this model defines a probability distribution on graphs of size $2n$.

For any graph $G = (V, E)$, consider any partition (V_+, V_-) of V, and let c be its characteristic function. Then for given parameters p and r, the following is the probability that G is generated from (V_+, V_-) in the way specified above.

$$\begin{aligned}&\Pr[\, G \text{ is generated from } (V_+, V_-)\,] \\ &\quad = \prod_{(i,j)\in E} p^{[c(i)=c(j)]} r^{[c(i)\neq c(j)]} \cdot \prod_{(i,j)\in \overline{E}} (1-p)^{[c(i)=c(j)]} (1-r)^{[c(i)\neq c(j)]}\end{aligned}$$

where $[\cdots]$ takes 1 if $\cdots$ holds and 0 otherwise. We call this probability the *likelihood*[2] w.r.t. (V_+, V_-). Now our main problem is defined as follows.

Most Likely Partition Problem
Input: An undirected graph $G = (V, E)$, and p and r, $0 < r < p < 1$.
Task: Find a partition V_+ and V_- of V with the max. likelihood w.r.t. p and r.
Remark. Find one of them if there are several solutions.

We will also consider a problem where parameters p and r are not given, which requires to compute also these parameters besides a partition. This harder version is called a *parameterless version*.

Dubhashi et al. gave [DLP03]. the following natural motivation to this problem (for bipartite graphs). Suppose that some set M of movies is classified into two groups, say, M^+, action movies, and M^-, romantic movies. Furthermore, a group P of people is also classified into two groups; P^+ and P^-, groups of those who prefer action (resp., romantic) movies. We assume that people go to see a movie following a simple probabilistic rule; for any $s \in \{+, -\}$, any one in P^s watches a movie in M^s with probability p, and he/she watches a movie in M^{-s} with probability r. Now a given a bipartite graph recording who watched which movie during some period (and also somehow given parameters p and r), we would like to obtain the classification of P and M. Then one natural solution is a partition (of P and M) so that the current record occurs most likely.

Unfortunately, it is easy[3] to prove that the Most Likely Partition problem is NP-hard; thus, one may not be able to hope for an efficient algorithm solving the problem for all instances.

[1] Throughout this paper, we will assume that the size of V is $2n$ in order to discuss the Graph Bisection problem. But this requirement is not necessary for our algorithms for the Most Likely Partition problem.

[2] The likelihood of (V_+, V_-) is, in short, $\Pr[(V_+, V_-)|G]$, whereas this probability is $\Pr[G|(V_+, V_-)]$. But since $\Pr[(V_+, V_-)|G] = \Pr[G|(V_+, V_-)] \cdot \Pr[G]/\Pr[(V_+, V_-)]$, and both $\Pr[G]$ and $\Pr[(V_+, V_-)]$ are the same for all possible partitions, we can use the probability $\Pr[(V_+, V_-)|G]$ for determining a most likely partition.

[3] One can prove the NP-hardness by showing a reduction from the MAX-CUT problem.

On the other hand, the problem is not so hard *on average* under some reasonable probability models on input graphs. A "planted solution model" is one of such models. A *planted solution model* is a way to generate input graphs, which is almost the same as the one explained above. Only one difference is that we fix an initial partition, which we call a *planted solution*, to be $V_+^* = \{1, ..., n\}$ and $V_-^* = \{n+1, ..., 2n\}$; that is, we assume that graphs are generated randomly from this partition. It can be shown [Ons05] that if n is large enough for $p-r$ (more specifically, $n = \Omega((p-r)^{-2})$, or equivalently $p - r = \Omega(n^{-1/2})$), then the planted solution is a unique solution for generated instances with high probability. Thus, such size parameter n, the goal of algorithms is to find the planted solution.

Although our algorithms are designed for the Most Likely Partition problem, the one for the parameterless version can be regarded as an algorithm for the following Graph Bisection problem.

Graph Bisection Problem
Input: An undirected graph $G = (V, E)$, where $|V| = 2n$ for some n.
Task: Find an equal size partition V_1 and V_2 of V with min. number of cut edges.
Remark. Find one of them if there are several solutions.

This problem has been studied extensively by many researchers [Bop87, Betal87, JS98, CK01], and several algorithms have been proposed. Although the problem is NP-hard [GJS76], these algorithms are shown to be efficient *on average* from computer experiments and/or mathematical analyses. For such average case analyses, the planted solution model has been often used, and here we follow this and assume the planted solution model. It has been shown [Bop87] that if n is large enough[4] for $p-r$ (more specifically, $n = \Omega((p-r)^{-2})$, or equivalently $p - r = \Omega(n^{-1/2})$), then the planted solution is a unique solution also for the Graph Bisection problem. Thus, for such size parameter n, we may regard our algorithm (for the parameterless version) as yet another algorithm for the Graph Bisection problem.

Our algorithms are derived from Pearl's belief propagation [Pea88] with some slight modification. Roughly speaking, belief propagation is a way to compute a marginal probability of the state of each node in a given Bayesian network. We use this technique for the Most Likely Partition problem. Consider any G, p, and r; these are given input instances for the Most Likely Partition problem. We first define a Bayesian network on which a belief propagation algorithm (in short, a BP algorithm) is expected to compute the following probability.

$$P(i) \;=\; \Pr[\, c(i) = +1 \,|\, G(V_+, V_-) = G \,], \tag{1}$$

Here the probability is defined under our random model for the Most Likely Partition problem. The approximation of $P(i)$ is in general called a *belief* (that i belongs to V_+). The BP algorithm computes beliefs in rounds; at each round, it updates beliefs and we would like to have correct $P(i)$'s at some round. In fact, it is shown that the BP algorithm converges in finite rounds and yields the correct probabilities if a given Bayesian network is a tree; although such a convergence

[4]This bound is of the same order as the one for the condition that a planted solution is with high probability a solution for the Most Likely Partition problem; but we do not know whether they are the same, or, if not, which bound is smaller.

cannot be guaranteed in general Bayesian networks, it is often the case that the BP algorithm converges and gives quite accurate values even for Bayesian networks with loops. Now suppose that the BP algorithm computes $P(i)$ correctly at some round, then a natural solution for our partition problem is to compute V_+ (resp., V_-) as a set of vertices i with $P(i) > 0.5$ (resp., $P(i) < 0.5$), which we may expect to give a partition with the max. likelihood. Let us call this algorithm a *BP-based (partition) algorithm*. This is the basis of our algorithms.

For many similar problems, this approach has been quite successful, and it is now one of the standard heuristics; see, e.g., [MMC98]. Note that it is not known whether such BP algorithms and/or BP-based algorithms work correctly for Bayesian networks with loops. It is one of the important problems to give rigorous justifications to those algorithms[5]. Our theoretical analysis is, though for quite limited usage of the belief propagation, one of few examples such that the belief propagation is justified rigorously.

Although our preliminary experiments show that the BP-based partition algorithm works quite well, we also found that some slight modification in the BP algorithm makes the whole algorithm more stable. Recall that the BP algorithm updates beliefs in rounds. The modification is simply to remove one restriction from belief propagation's rule for updating beliefs. More specifically, we do not exclude any incoming message for updating beliefs; see, the next section for the detail. Clearly, for each vertex i, a value computed by this modified algorithm is no longer a belief that the original BP computes; thus, we will call it a *pseudo belief*. Nevertheless, we found that with this modification both convergence probability and speed are improved, and hence, the partition algorithm becomes more stable. (The accuracy of a solution obtained from a convergent state is equally high for both algorithms.)

Some theoretical analysis is made on this partition algorithm. More precisely, we consider the following version that we denote `PartByPseudoBP2`; the algorithm updates, in its BP part, beliefs[6] only *twice*, and then determine a partition based on the obtained beliefs. This algorithm can be implemented so that it runs in $\mathcal{O}(n+m)$ steps. For this algorithm, we prove the following.

Theorem 1. For any n, and p and r, $0 < r < p < 1$, consider the execution of `PartByPseudoBP2` on randomly generated graphs under the planted model. Then we have

$$\Pr[\text{ the algorithm yields the planted solution }] \;\geq\; 1 - 2n \cdot e^{-\epsilon_1 n \cdot \frac{(p-4)^4}{p^2}},$$

where ϵ_1 is some constant.

For any $\delta > 0$ and any p and r, $0 < r < p < 1$, consider any $n = \Omega(p^2(p-r)^{-4}\log(1/(\delta(p-r))))$. As mentioned above, one can show that the probability that the planted solution is not a correct answer is small, say, $< \delta/2$; that is, so long as the algorithm gives the planted solution, it should be the correct answer with hight probability. Hence, it follows from the above that the success probability, the probability that the algorithm yields a correct answer, is larger than $1 - \delta$.

The accuracy can be explained in terms of the bound for $p - r$, which is more useful for comparing with the other algorithms. Here we consider only the simplest case where p is of the

[5]Precisely speaking, they should be called "heuristics" instead of "algorithms" until rigorous justification is given.

[6]In spite of our modification, since the number of rounds of belief updates is two, the computation of the algorithm `PartByPseudoBP2` is essentially the same as that of the original.

same order of $p - r$. In this case, we can show that the algorithm yields a correct answer with probability $> 1 - \delta$ if $p - r = \Omega(n^{-1/2} \log(n/\delta))$. The performance of the algorithm is not so good as this if p is large and $p - r$ is small.

For the Graph Bisection problem, we need to solve the parameterless version. Note here that we may assume that a planted partition is of equal size; hence, we can estimate $p + r$ by counting the number of edges in a given graph, and we can expect that this estimated value can be quite accurate. Then within some appropriate range, we search for approximations $\widetilde{p}$ and $\widetilde{r}$ of p and r by binary search. This needs to run the algorithm `PartByPseudoBP2` for $\mathcal{O}(\log n)$ times (if $p - r = \Omega(n^{-1/c})$ for some $c > 1$). The accuracy of this algorithm is guaranteed by the following theorem.

Theorem 2. For any n, and p and r, $0 < r < p < 1$, consider the execution of `PartByPseudoBP2` on randomly generated graphs under the planted model, but here execute it with parameters $\widetilde{p}$ and $\widetilde{r}$ such that $\widetilde{p} + \widetilde{r} = p + r$ and $p - r \leq \widetilde{p} - \widetilde{r} < (5/4)(p - r)$. Even in this case we have

$$\Pr[\text{ the algorithm yields the planted solution }] \;\geq\; 1 - 2n \cdot e^{-\epsilon_2 n \cdot \frac{(p-r)^4}{p^2}},$$

where ϵ_2 is some constant.

Although the above theoretial analyses guarantee the performance for very limited versions, our preliminary computer experiments show that, for the Most Likely Partition problem, the more general version (allowing more belief update rounds) performs well for much larger range of $p - r$. In fact, even for the case $p - r = o(n^{-1/2})$ (though obviously, there is some limit), this general algorithm seems to find a most likely partition, which is not necessary the planted solution.

1.1 Related Works

The Graph Bisection problem has been studied by many researchers; many heuristics and algorithms have been proposed, and some of them have been analyzed mathematically. One of the first mathematical analysis of the average-case performance of bisection algorithms is due to Bui et al in [Betal87]. Later Dyer and Frieze [DF89] gave an algorithm that works for dense graphs. Boppana [Bop87] gave an algorithm based on the ellipsoid method, which is now shown to work on the largest class of instances among existing (expected) polynomial-time algorithms. Unfortunately, however, though it is polynomial-time, the time complexity of Boppana's algorithm is not so low. Much simpler and similarly effective algorithms have been proposed more recently by Condon and Karp [CK01] for general graphs.

The average-case analysis of partitioning algorithms on the the planted solution model used in this paper was introduced by Jerrum and Sorkin [JS98], where they proved that an algorithm based on the simulated annealing heuristics works well for the Graph Bisection problem. Their algorithm finds a planted solution in $\mathcal{O}(n^{2+\epsilon})$ with high probability if $p - r > n^{-1/6}$. Later Condon and Karp gave a simpler algorithm and proved that it finds a planted solution in linear time if $p - r > n^{-1/2+\epsilon}$ with probability $1 - \exp(-n^{\Theta(\epsilon)})$. Dubhashi et al [DLP03] proposed some nice senario for studying graph partition problems under the planted solution, and they gave anoher simple algorithm that solves the Graph Bisection problem for bipartite graphs.

program PseudoBP($G = (V, E)$, p, r);
% This algorithm is for computing a pseudo belief
% (in approx. lex) to x_i for each vertex $i \in V$.
% W.l.o.g. we assume $c(1) = +1$.
begin
  set all x_i to 0;
  repeat MAXSTEP times **do** {
    $x_1 \leftarrow +\infty$;
    for each $i \in V$,
    $x_i \leftarrow \sum_{j \in N(i)} h_+ \cdot \mathrm{Th}_+(x_j) - \sum_{j \notin N(i)} h_- \cdot \mathrm{Th}_-(x_j)$;
    if all beliefs are stabilized **then break**;
  }
  output($x_1, ..., x_{2n}$);
  % Classification is $(\mathrm{sg}(x_1), ..., \mathrm{sg}(x_{2n}))$.
end-program

parameters & functions

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

Figure 1: Computation for pseudo beliefs (by approximate lex)

Compared with the above algorithms, our algorithm, e.g., PartByPseudoBP2, has the following features: Firstly it is deterministic and runs in $\mathcal{O}(n + m)$ time. All the other algorithms are randomized except for Boppana's algorithm, and Boppana's algorithm has much higher complexity. Secondly it is applicable to wider ranges of $p - r$, compared with the algorithm of Dyer and Frieze and the simulated annealing based algorithm of Jerrum and Sorkin. The range for $p - r$ that our theorems for PartByPseudoBP2 guarantees is slightly smaller than the one for the algorithm of Condon and Karp. On the other hand, our preliminary experiments show that a more general version (allowing more belief update rounds) seems to perform much better than the other algorithms.

2 Our Base Algorithm

In Figure 2 we state our base algorithm; for a given graph $G = (V, E)$ and parameters p and r, it computes as x_i a "pseudo belief" for each vertex $i \in V$. It should be remarked here that this "pseudo belief" is different the one explained in the previous section, but the approximation of its log value that we will explain below; but anyway, the value x_i is somehow related to the "belief" that $c(i) = +1$ with the most likely classification c. A natural interpretation of this belief is that $c(i) = +1$ if and only if x_i is positive. A partition algorithm — PartByPseudoBP — is an algorithm that outputs this classification after pseudo beielfs are computed by PseudoBP. In the following, we will simply use "belief" for this "pseudo belief".

Some explanation may be necessary. Without losing generality, we may assume that the first vertex (i.e., vertex 1) is positive by our target classification c. Hence, x_1 is set $+\infty$ at the beginning of each repetition. Let $N(i)$ be the set of vertices having an edge with i in G. We say that the belief of vertex i exceeds the threshold if x_i is not in a range $[-\mathrm{th}_+, +\mathrm{th}_+]$. (Note that

$\mathrm{th}_- < \mathrm{th}_+$; hence, this range is larger than $[-\mathrm{th}_-, +\mathrm{th}_-]$.) By "round" we mean the number of repeat iterations. We use a parameter MAXSTEP to bound the number of iterations; in particular, the one using MAXSTEP = 2, denoted as `PartByPseudoBP2`, will be analyzed in the next section.

A Remark on the Implementation of the Algorithm

For general `PseudoBP`, we need $\mathcal{O}(n^2)$ floating point operations for computing x_i at each round. It is, nevertheless, possible to implement the special case `PartByPseudoBP2` so that the computation of x_i's at each round is executed in $\mathcal{O}(n+m)$ steps. For computing x_i for each vertex i, we simply have to count the number of edges between i and vertices in $N(1)$ and vertices not in $N(1)$. (Recall that $N(1)$ is the set of neighbors of the vertex 1.) This counting for all x_i's can be done in $\mathcal{O}(n+m)$ steps, if an input graph is given by some appropriate data structure.

2.1 A Derivation of the Algorithm

Our base algorithm `PseudoBP` is derived from the standard belief propagation algorithm for computing the marginal probabilities $P(i)$ defined by (1). We briefly explain this derivation. Below we follow [MMC98] for notions and notations on the belief propagation. (Although we will not explain the precise meaning of such notations, it is not essential for our later discussion.)

Let $G = (V, E)$ be a given graph with $2n$ vertices. For any $i, j \in V$, we let $e_{ij} = +1$ if there exists an edge between i and j in E, and $e_{ij} = -1$ otherwise. A Bayesian network for G is a graph consisting of nodes $\{N_i\}_{i \in V}$ corresponding to all vertices in V nodes $\{Z_{ij}\}_{(i,j) \in V \times V}$ corresponding to all pairs in $V \times V$. The belief propagation updates beliefs on these nodes by exchanging messages between them. But since those messages are quite simple in our case, we can simplify this scheme so that messages are exchanged between nodes corresponding to vertices in V. For each pair of vertices $i, j \in V$, two messages $\pi_{ij}(x)$, $x \in \{-1, +1\}$, sent from node N_i to N_j are computed as follows from the messages π_{ki} that node N_i received at the previous round.

$$\pi_{ij}(x) \;=\; \alpha q_i(x) \prod_{k \neq j} (\delta_{ij}(r)\pi_{ki}(-x) + \delta_{ij}(p)\pi_{ki}(x)), \tag{2}$$

Here q_i, α, and δ_{ij} have the following meaning: $q_i(x)$ is a priori probability of $c(i) = x$ (in our case, $q_i(+1) = q_i(-1) = 1/2$ except for the vertex 1); α is a normalization factor to keep $\pi_{ij}(+1) + \pi_{ij}(-1) = 1$; and $\delta_{ij}(y) = y$ if $e_{ij} = +1$, and $\delta_{ij}(y) = 1 - y$ otherwise. Notice here that for computing a message π_{ij} from node i to node j, the previous value of π_{ji}, a message from node j, is not used. This is the point we will relax in our modification. A belief b_i at node N_i, intuitively the belief for $c(i) = +1$, is then computed as

$$B_i \;=\; \frac{\prod_{j \in V} \pi_{ji}(+1)}{\prod_{j \in V} \pi_{ji}(+1) + \prod_{j \in V} \pi_{ji}(-1)}.$$

Now we make several simplifications for our problem. First in order to reduce the number of variables, we use $m_{ij} = \pi_{ij}(+1)/\pi_{ij}(-1)$ and $b_i = \prod_{j \in V} m_{ji}$; also let $\rho_i = q_i(+1)/q_i(-1)$. Note that we can now consider $c(i) = +1$ if $b_i > 1$ and $c(i) = -1$ if $b_i < 1$. The following updating rule is obtained from (2).

$$m_{ij} \;=\; \rho_i \prod_{k \neq j} f_{ij}(m_{ki}),$$

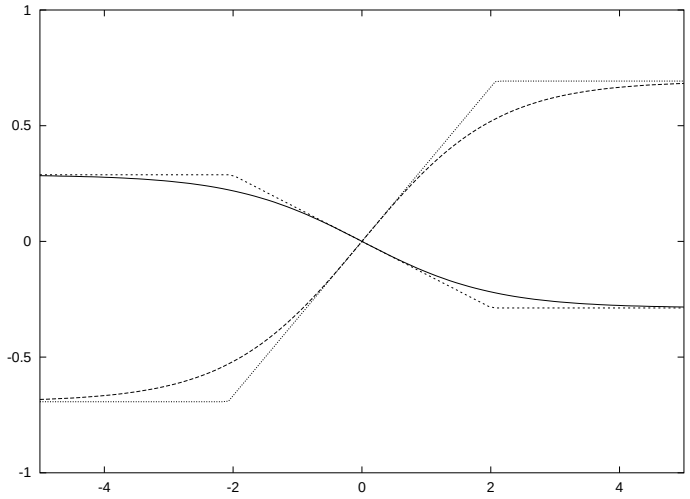


Figure 2: lex_+ and lex_- and their approximations $\widetilde{\mathrm{lex}}_+$ and $\widetilde{\mathrm{lex}}_-$ (for $p = 0.4$ and $r = 0.2$)

where $f_{ij}(x)$ is defined by

$$f_{ij}(x) \;=\; \frac{a_{ij}x+1}{x+a_{ij}}, \qquad a_{ij} \;=\; \begin{cases} a_+ = \frac{p}{r}, & \text{if } e_{ij} = +1, \text{ and} \\ a_- = \frac{1-p}{1-r}, & \text{if } e_{ij} = -1. \end{cases}$$

At this point, we introduce one a priori knowledge. Without losing generality, we may fix the classification of one vertex; for example, let us assume that vertex 1 belongs to V_+, i.e., $c(1) = +1$. This means that $q_1(+1) = 1$ and $q_1(-1) = 0$, implying that $\rho_1 = +\infty$ and $m_{1j} = +\infty$. For the other i's, we have $q_i(+1) = q_i(-1) = 0.5$, and hence, $\rho_i = 1$. Thus, we have the following simplified rule.

$$m_{1j} \;=\; +\infty, \;\text{ and }\; m_{ij} \;=\; \prod_{k\neq j} f_{ij}(m_{ki}).$$

Here we may define $f_{ij}(+\infty) = a_{ij}$ and $f_{ij}(-\infty) = 1/a_{ij}$.

Let us convert this updating rule to additive one. For this purpose, we introduce $\ell_{ij} = \ln(m_{ij})$ and a function lex defined by

$$\mathrm{lex}_{ij}(x) = \ln(f_{ij}(e^x)). \tag{3}$$

Then we have, for all $i, j \in V$, $i \neq 1$,

$$\ell_{ij} \;=\; \sum_{k\neq j} \mathrm{lex}_{ki}(\ell_{ki}), \tag{4}$$

Note that $\ell_{1j} = +\infty$. The logarithmic belief $\ln(b_i)$ is computed as $\sum_{j\in V} \ell_{ij}$, and $c(i)$ is determined whether it is positive or negative.

We simplify the above computation a bit further. As shown in Figure 2.1, both functions lex_+ and lex_- can be approximated well by some linear functions with thresholds. More specifically, we consider the following functions for approximating lex_σ, $\sigma \in \{+, -\}$.

$$\widetilde{\mathrm{lex}}_\sigma(x) \;=\; \begin{cases} h_\sigma \cdot \mathrm{th}_\sigma, & \text{if } \mathrm{th}_\sigma < x, \\ \sigma h_\sigma \cdot x, & \text{if } -\mathrm{th}_\sigma \le x \le \mathrm{th}_\sigma, \text{ and} \\ -h_\sigma \cdot \mathrm{th}_\sigma, & \text{if } x < \mathrm{th}_\sigma, \end{cases} \tag{5}$$

where h_σ (i.e., h_+ and h_-) and th_σ (i.e., th_+ and th_-) are those defined in Figure 2. Our (linearlized version of) belief propagation algorithm is to compute messages by (4) with these approximations of lex functions.

Finally we introduce one modification. When computing a message from node i to node j, we do not exclude the previous value of w_{ji}, a message that came from j to i. Then there is no distinction between messages to j and to j', and we only need to consider the following quantity

$$x_i \;=\; \sum_{k \in V} \widetilde{\mathrm{lex}}_{ki}(x_k), \tag{6}$$

which we may interpret as a message from i to any node. Furthermore, we may now consider it also as a quantity corresponding to $\ln(b_i)$, which we will call a *pseudo belief*. It is easy to see that our base algorithm `PseudoBP` computes this pseudo belief by using the updating formula (6).

3 Analysis of `PartByPseudoBP2`

Here we prove two theorems stated in Introduction. From our base algorithm `PseudoBP`, by using setting MAXSTEP = 2 and modifying it to output classification instead of beliefs, we obtain the algorithm `PartByPseudoBP2` that we will study in this section. But in order to simplify our analysis, we modify the algorithm a bit more[7]: (i) in order to avoid some difficulty from applying the thresholding, use some small value $\theta < \min(\mathrm{th}_+, \mathrm{th}_-)$ for the inivial value of x_1, (ii) at the second belief update round, simply set $x_1 = 0$, thereby ignoring the effect from the vertex 1, and (iii) the output classification vertex 1 is $+1$ no matter what value is computed at x_1. Precisely speaking, this is the algorithm `PartByPseudoBP2` we will investigate below.

Now we prove Theorem 1. Below let $G = (V, E)$ be a random graph of size $|V| = 2n$ generated from the planted solution $V^*_+ = \{1, ..., n\}$ and $V^*_- = \{n+1, ..., 2n\}$ with parameters p and r, $0 < r < p$. Let c^* be its characteristic function. For any $i, j \in V$, let $E_{i,j}$ be a random variable taking a value in $\{0, 1\}$ that indicates whether there exists an edge between vertices i and j in G; hence, $E_{i,j} = 1$ with prob. p if $c^*(i) = c^*(j)$, and otherwise $E_{i,j} = 1$ with prob. r. Define $\alpha = p - r$ and $\beta = p + r$, which will be used in the following analysis.

Consider the execution of the algorithm. Let us introduce some notations. We use x_i to denote the value computed at the variable x_i after the execution of the algorithm. On the other hand, the value of the variable x_i after the first round is denoted as $x_i^{(1)}$. Since we treat the vertex 1 separately, by V^*_+, we simply denote the set $\{2, ..., n\}$ ignoring the vertex 1. We introduce random variables (where the randomness is due to the random graph G). Let P_1 and P_0 be the set of vertices in V^*_+ that respectively does/does not have an edge with the vertex 1; sets N_1 and N_0 are defined similarly for V^*_-. It is easy to see that $x_i^{(1)} = h_+ \cdot \theta$ if $i \in P_1 \cup N_1$ and $x_i^{(1)} = -h_- \cdot \theta$ if $i \in P_0 \cup N_0$. Recall that θ is an initial value for x_1, which is introduced in place of ∞ for avoiding technical difficulty from the thresholding. Since θ is a common factor for all the following values, we omit denoting θ in the following discussion.

[7] We think that the modifications (i) and (ii) are not essential; but it may be possible that more detail calculation without any modification leads some small improvement.

We further introduce the following random variables.

$$\begin{array}{rclcrcl} Y^+ & = & |P_1|, & & Y^- & = & |N_1|, \\ X_i^{+,1} & = & \sum_{j\in P_1} E_{i,j}, & & X_i^{+,0} & = & \sum_{j\in N_1} E_{i,j}, \\ X_i^{-,1} & = & \sum_{j\in P_0} E_{i,j}, & & X_i^{-,0} & = & \sum_{j\in N_0} E_{i,j}. \end{array}$$

On these variables, we have the following two claims.

Claim 1. For any $i \in V - \{1\}$, we have

$$\begin{aligned} x_i \;=\; & 2n\cdot h_-^2 - h_-(h_+ + h_-)(Y^+ + Y^-) \\ & + h_+(h_+ + h_-)(X_i^{+,1} + X_i^{-,1}) - h_-(h_+ + h_-)(X_i^{+,0} + X_i^{-,0}). \end{aligned} \tag{7}$$

Proof. It is easy to see that x_i is calculated as follows, from which the claim follows immediately.

$$\begin{aligned} x_i \;=\; & \sum_{j\in P_1} \big(E_{ij}h_+^2 + (1-E_{ij})(-h_-)h_+\big) + \sum_{j\in P_0} \big(E_{ij}h_+(-h_-) + (1-E_{ij})(-h_-)^2\big) \\ & + \sum_{j\in N_1} \big(E_{ij}h_+^2 + (1-E_{ij})(-h_-)h_+\big) + \sum_{j\in N_0} \big(E_{ij}h_+(-h_-) + (1-E_{ij})(-h_-)^2\big) \end{aligned}$$

□

Claim 2. For any $i \in V - \{1\}$, we have

$$\mathrm{E}[x_i] \;=\; c^*(i)\cdot \frac{2n\beta^4}{\alpha^2(2-\alpha)^2}. \tag{8}$$

Proof. The proof is tedious but easy caliculation, once we have the following expectations.

$$\begin{array}{rclcrcl} \mathrm{E}[Y^+] & = & pn, & & \mathrm{E}[Y^-] & = & rn, \\ \mathrm{E}[X_i^{+,1}] & = & \begin{cases} p^2n & \text{if } i\in V_+, \text{ and} \\ prn & \text{if } i\in V_-, \end{cases} & & \mathrm{E}[X_i^{+,0}] & = & \begin{cases} (1-p)pn & \text{if } i\in V_+, \text{ and} \\ (1-p)rn & \text{if } i\in V_-, \end{cases} \\ \mathrm{E}[X_i^{-,1}] & = & \begin{cases} r^2n & \text{if } i\in V_+, \text{ and} \\ prn & \text{if } i\in V_-, \end{cases} & & \mathrm{E}[X_i^{-,0}] & = & \begin{cases} (1-r)rn & \text{if } i\in V_+, \text{ and} \\ (1-r)pn & \text{if } i\in V_-, \end{cases} \end{array}$$

Although these are again easy to obtain, there are some points that we need to be a bit careful. For example, for any $i \in V_+$, the expectation $\mathrm{E}[X_i^{+,1}]$, i.e., the expectation of $X_i^{+,1} = \sum_{j\in P_1} E_{ij}$ seems $\mathrm{E}[\,|P_1|\,]\cdot\Pr[E_{ij}=1] = pn\cdot p$. This is in fact true due to the independence of $Y^+ = |P_1|$ and E_{ij}; let us check this for the sake of completeness.

Let $i \in V_+$. For any set $U \subseteq V$, and any $j \in V$, we have $\Pr[\,E_{ij}=1 \,|\, P_1 = U\,] = p$, which is due to independence of E_{ij} and E_{1k} for all $k \in V$. Hence, we have $\mathrm{E}[\,X_i^{+,1} \,|\, P_1 = U\,] = p\cdot|U|$; in other words, for any $y \geq 0$, we have $\mathrm{E}[\,X_i^{+,1} \,|\, Y^+ = y\,] = py$. Then we have $\mathrm{E}[\,X_i^{+,1}\,] = \sum_y \Pr[\,Y^+ = y\,]\cdot\mathrm{E}[\,X_i^{+,1} \,|\, Y^+ = y\,] = p\cdot\sum_y \Pr[\,Y^+ = y\,]\cdot y = p\cdot\mathrm{E}[\,Y^+\,] = p(pn)$. □

The above claim shows that each x_i gives *on average* the correct classification. Thus, it now suffices to show a condition that all x_i's are close to their expectations. In the following, consider any $i \in V_+^* - \{1\}$, and we show a sufficient condition for x_i being positive. Furthermore, we consider the case where $\alpha\ (= p + r) \leq 1$. (The case $\alpha > 1$ and the case $i \in V_-^*$ can be analyzed similarly (in fact, even by slightly easier arguments), and the analysis for these cases is omitted.)

For any δ_+, δ_-, γ_+, γ_-, γ'_+, and γ'_-, all taking values in $[0,1)$, consider the following situation.

$$\begin{aligned}
Y^+ &= \mathrm{E}[Y^+] + \delta_+ n = pn + \delta_+ n,\\
Y^- &= \mathrm{E}[Y^-] + \delta_- n = rn + \delta_- n,\\
X_i^{+,1} &= (pn + \delta_+ n)(p - \gamma_+) = p^2 n + (p\delta_+ - p\gamma_+ - \delta_+\gamma_+)n,\\
X_i^{-,1} &= (rn + \delta_- n)(r - \gamma_-) = r^2 n + (r\delta_- - r\gamma_- - \delta_-\gamma_-)n,\\
X_i^{+,0} &= ((1-p)n - \delta_+ n)(p + \gamma'_+) = (1-p)pn + ((1-p)\gamma'_+ - p\delta_+ - \delta_+\gamma'_+)n, \text{ and}\\
X_i^{-,0} &= ((1-r)n - \delta_- n)(r + \gamma'_-) = (1-r)rn + ((1-r)\gamma'_- - r\delta_- - \delta_-\gamma'_-)n.
\end{aligned}$$

We may also consider the other situations such as the case where $Y^+ = \mathrm{E}[Y^+] - \delta_+ n$; but it is easy to check that the above choice makes x_i the smallest (under the assumption that $\alpha \le 1$).

Evaluate (7) with these expectations; by easy but tedious caliculation, we have

$$\begin{aligned}
x_i \;=\; \mathrm{E}[x_i] \;&- nh_-(h_+ + h_-)\,(\,\delta_+ + \delta_-\,)\\
&- nh_-(h_+ + h_-)((1-p)\gamma'_+ + (1-r)\gamma'_-)\\
&- nh_+(h_+ + h_-)\,(\,p\gamma_+ + r\gamma_- + \delta_+\gamma_+ + \delta_-\gamma_-\,)\\
&+ nh_+(h_+ + h_-)\,(\,p\delta_+ + r\delta_-\,)\\
&+ nh_-(h_+ + h_-)\,\big(\,p\delta_+ + r\delta_- + \delta_+\gamma'_+ + \delta_-\gamma'_-\,\big)\,.
\end{aligned} \tag{9}$$

Here we may let $\gamma'_+ = \gamma'_- = \gamma$ and ignore the last positive term, which implies

$$\begin{aligned}
x_i \;\ge\; \mathrm{E}[x_i] \;&- n(h_+ + h_-)\,(\,h_-(\delta_+ + \delta_-) - h_+(p\delta_+ + r\delta_-)\,)\\
&- nh_-(h_+ + h_-)(2-\alpha)\gamma \;-\; nh_+(h_+ + h_-)\,(\,p\gamma_+ + r\gamma_- + \delta_+\gamma_+ + \delta_-\gamma_-\,)\,.
\end{aligned}$$

By expressing h_+ and h_- by α and β, we have

$$\begin{aligned}
x_i \;\ge\; \mathrm{E}[x_i] \;&- \frac{2n\beta^3(\delta_- - \delta_+)}{\alpha^2(2-\alpha)^2} \;-\; \frac{2n\beta^2}{\alpha(2-\alpha)^2}\,(\,p\delta_+ + r\delta_-\,)\\
&- \frac{2n\beta^2\gamma}{\alpha(2-\alpha)} \;-\; \frac{2n\beta^2}{\alpha^2(2-\alpha)}\,(\,p\gamma_+ + r\gamma_- + \delta_+\gamma_+ + \delta_-\gamma_-\,)
\end{aligned}$$

From this we immediately have the following claim.

Claim 3. For any $i \in V_+ - \{1\}$, assume the following bounds hold for the parameters defined above.

$$\begin{aligned}
\delta_+ &< \min\left(p,\, \frac{\beta^2}{8p\alpha}\right), \quad \delta_- < \min\left(p,\, \frac{\beta}{8},\, \frac{\beta^2}{8r\alpha}\right),\\
\gamma_+ &< \frac{\beta^2}{8p(2-\alpha)}, \qquad \gamma_- < \frac{\beta^2}{8p(2-\alpha)}, \qquad \text{and} \qquad \gamma < \frac{\beta^2}{8\alpha(2-\alpha)}.
\end{aligned}$$

Then we have $x_i > \mathrm{E}[x_i] - 2n\beta^4/(\alpha^2(2-\alpha)^2) = 0$.

Now for the theorem it suffices to show that the bounds in the above claim hold with probability $1-\delta$ if n is large enough. We use the following Chernoff bound.

Proposition 1. Let $X_1, ..., X_m$ be m the outcomes of independent Bernoulli trials, i.e., independent random variables with $\Pr[X_i = 1] = q$ and $\Pr[X_i = 0] = 1 - q$. Then there exists some constant $c_0 > 1$ such that for any ε, we have

$$\Pr[\,\sum_{1\le i\le m} X_i \ge (1+\varepsilon)qm\,] \;\le\; e^{-mq\varepsilon^2/c_0}, \quad \text{and} \quad \Pr[\,\sum_{1\le i\le m} X_i \le (1-\varepsilon)qm\,] \;\le\; e^{-mq\varepsilon^2/c_0}.$$

Claim 4. There exists some constant ϵ_1 such that for any p, r, $0 < r < p < 1$ (here considering the case that $\alpha = p + r \leq 1$), we have

$$\Pr[\text{ all bounds of Claim 3 hold }] \;\geq\; 1 - e^{-\epsilon_1 n \cdot \frac{\beta^4}{p^2}}. \tag{10}$$

Proof. Recall that, e.g., $Y^+ = |P_1| = \sum_{j \in V_+^*} E_{1,j}$ and $X_i^{+,1} = \sum_{j \in P_1} E_{i,j}$, and that $E_{i,j}$ are independent Bernoulli trials. Thus, roughly speaking, for each bound in Claim 3, we use the Chernoff bound to show that the probability that the bound *does not* hold is at most $e^{-\epsilon_1 n(\beta^4/p^2)}/8$, thereby deriving the bound of the claim. But since some random variables, e.g., $X_i^{+,1} = \sum_{j \in P_1} E_{i,j}$, depend on the others, e.g., P_1, we need to be a bit careful.

First consider the bounds for δ_+ and δ_-. For these, we can simply apply the Chernoff bound. For example, since $Y^- = \sum_{j \in V_-^*} E_{1,j}$, $\Pr[\, E_{1,j} = 1 \,] = r$ for any $j \in V_-^* = \{n+1, ..., 2n\}$, and $\alpha \leq 1$, we have

$$\begin{aligned}
&\Pr[\, \delta_- \geq p \,] \;=\; \Pr[\, Y^- \geq \left(1 + \frac{p}{r}\right) rn \,] \;\leq\; e^{-nr \cdot (p/r)^2/c_0} \;=\; e^{-n(p^2/r)/c_0}, \\
&\Pr\left[\, \delta_- \geq \frac{\beta}{8} \,\right] \;=\; \Pr\left[\, Y^- \geq \left(1 + \frac{\beta}{8r}\right) rn \,\right] \;\leq\; e^{-nr \cdot (\beta/8r)^2/c_0} \;=\; e^{-n(\beta^2/64r)/c_0}, \text{ and} \\
&\Pr\left[\, \delta_- \geq \frac{\beta^2}{8p\alpha} \,\right] \;\leq\; \Pr\left[\, Y^- \geq \left(1 + \frac{\beta^2}{8pr}\right) rn \,\right] \;\leq\; e^{-nr \cdot (\beta^2/8pr)^2/c_0} \;=\; e^{-n(\beta^2/64p^2r)/c_0}.
\end{aligned}$$

Hence, to show that these are all smaller than $e^{-\epsilon_1 n(\beta^4/p^2)}/(3 \cdot 8)$ for some constant ϵ_1, we only have to confirm the following.

$$\frac{p^2}{r},\; \frac{\beta^2}{r},\; \frac{\beta^2}{p^2 r} \;\geq\; \frac{\beta^4}{p^2}.$$

Next consider the other bounds, i.e., the bounds for γ_+, γ_-, and γ. Since the bound for γ_- is most subtle, we explain our argument for bounding the probability that $\gamma_- \geq \beta^2/(8p(2-\alpha))$ holds; below let $\epsilon = \beta^2/(8p(2-\alpha))$. Here we consider this probability under the condition that $Y^- = rn + \delta_- n$ for some $\delta_- \geq 0$. More precisely, under the condition that $P_0 = U$ for some $U \subseteq V_-^*$ such that $|U| = rn + \delta_- n$. Then since $X_i^{-,1} = \sum_{j \in U} E_{ij}$ and these E_{ij}'s are mutually independent and independent from the choice of U, we can use the Chernoff bound to show

$$\begin{aligned}
&\Pr[\, \gamma_- \geq \epsilon \,|\, P_0 = U \,] \\
&\quad= \;\Pr[\, X_i^{-,1} \,(= (rn + \delta_- n)(r - \gamma_-)) \leq r((r+\delta_-)n) - \epsilon((r+\delta_-)n) \,|\, P_0 = U \,] \\
&\quad= \;\Pr[\, X_i^{-,1} \leq (1 - \epsilon/r) r((r+\delta_-)n) \,|\, P_0 = U \,] \\
&\quad\leq\; e^{-((r+\delta_-)n) r (\epsilon/r)^2/c_0} \;\leq\; e^{-nr^2(\epsilon/r)^2/c_0} \;=\; e^{-n\epsilon^2/c_0}.
\end{aligned}$$

Since $\epsilon = (\beta^2/p) \cdot (1/8(2-\alpha) \geq (\beta^2/p) \cdot (1/16)$, we have $\Pr[\, \gamma_- \geq \epsilon \,|\, P_0 = U \,] \leq e^{-\epsilon_1 n(\beta^4/p^2)}/8$ for some constant ϵ_1. Note that this bound holds uniformly for any $U \subseteq V_-^*$ such that $|U| = rn + \delta_- n$; hence the same bound holds for $\Pr[\, \gamma_- \geq \epsilon \,|\, Y^- = rn + \delta_- n \,]$.

Now, for example, for $\epsilon' = \min(p, \beta/8, \beta^2/(8r\alpha))$, we have

$$\begin{aligned}
&\Pr[\, \gamma_- < \epsilon \wedge \delta_- < \epsilon' \,] \\
&\quad= \;\Pr[\, \gamma_- < \epsilon \,|\, Y^- = rn + \delta_- n \wedge \delta_- < \epsilon' \,] \cdot \Pr[\, \delta_- < \epsilon' \,] \\
&\quad\leq\; \left(1 - e^{-\epsilon_1 n(\beta^4/p^2)}/8\right) \cdot \left(1 - e^{-\epsilon_1 n(\beta^4/p^2)}/8\right) \\
&\quad\leq\; 1 - 2e^{-\epsilon_1 n(\beta^4/p^2)}/8
\end{aligned}$$

By similar arguments, we can give the desired bound (10) for the probability that all bounds of Claim 3 hold. □

Next we consider Theorem 2. Before proving the theorem, we clarify our algorithmic strategy and recall the motivation of the theorem.

For the parameterless version, the goal of the problem is to compute, for a given graph G, parameters p and r and a partition to achieve the maximum likelihood. But here we assume that input graphs are generated under the planted solution model, and relax our goal to obtain a planted solution and compute parameters p' and r' close enough[8] to those used to generate the input graph from the planted solution. For this goal, we consider the following algorithm: First by counting the number of edges, we compute the estimation $\widetilde{\alpha}$ of α $(= p + r)$, which should be very close to the real value $p + r$ if n is sufficiently large. Then by using a guess $\widetilde{\beta}$ of β, run the algorithm `PartByPseudoBP2` with guessed $\widetilde{p}$ and $\widetilde{r}$. (Note that $\widetilde{p} = (\widetilde{\alpha} + \widetilde{\beta})/2$ and $\widetilde{r} = (\widetilde{\alpha} - \widetilde{\beta})/2$.) The initial value of $\widetilde{\beta}$ is the largest candidate, i.e., $\widetilde{\alpha}$; that is the case where $\widetilde{p} = \widetilde{\alpha}$ and $\widetilde{r} = 0$. If any "consistent partition" is obtained, then output the partition. Otherwise, revise $\widetilde{\beta}$ with $(4/5)\widetilde{\beta}$, and repeat the above process. The consistency of the partition can be checked as follows: First check whether it is an equal size partition (because we assume the planted solution model). Next estimate parameters p' and r' by couting the number of edges to vertices in the same set and in the other set, and confirm that (i) the estimation does not differ so much between vertices, and (ii) the same partition is obtained by running the algorithm with p' and r'. (Note that these p' and r', which may be different from $\widetilde{p}$ and $\widetilde{r}$ used to obtain the partition, are better approximations of p and r.)

For justifying the above algorithm, we only need to show the following property: a planted solution can be obtained with high probability if $\widetilde{\alpha}$ is estimated accurate enough and the guessed $\widetilde{\beta}$ satisfies $\beta \leq \widetilde{\beta} < (5/4)\beta$. So long as $\beta = \Omega(n^{-1/c})$ for some $c > 1$, $\widetilde{\beta}$ satisfying this condition can be obtained in $\mathcal{O}(\log n)$ iterations. For simplifying our discussion, we assume here that $\widetilde{\alpha}$ is estimated exactly, i.e., $\widetilde{\alpha} = \alpha$ $(= p + r)$, and gives a sufficient condition such that the above property holds. This is our Theorem 2. (The effect of the error $|\widetilde{\alpha} - \alpha|$ can be analyzed in the same way as the above proof of Theorem 1; we can derive some error bound and show some sufficient condition for n, which is similar to the one for Theorem 1. We leave this analysis to the reader.)

Now we prove Theorem 2. In the following, let $\widetilde{p}$ and $\widetilde{r}$ be those obtained from guessed $\widetilde{\beta}$ of β. Since we assume that $\widetilde{\alpha} = \alpha$, we have $\widetilde{p} + \widetilde{r} = p + r$. Define Δ so that $\widetilde{p} = p + \Delta$; from our assumption, we have $\widetilde{r} = r - \Delta$ and $\widetilde{\beta} = \beta + 2\Delta$. For the theorem, we may assume that $0 \leq \Delta < \beta/8$. But for a while, let us also consider the case where $\Delta < 0$, i.e., the case where we underestimate β.

Consider any $i \in V - \{1\}$. Here we use the same notations used in the proof of Theorem 1. Let $\widetilde{\mathrm{E}}[x_i]$ denote the value of the righthand expression of (8) computed with $\widetilde{p}$ and $\widetilde{r}$ for p and r; that is, it is the expectation of x_i if $\widetilde{p}$ and $\widetilde{r}$ were the parameters for generating the random input graph. Then we can reuse the formula (9); the difference $x_i - \mathrm{E}[x_i]$ computed here is the same as $\mathrm{E}[x_i] - \widetilde{\mathrm{E}}[x_i]$ with $\delta_+ = \Delta$, $\delta_- = -\Delta$, $\gamma_+ = -\Delta$, $\gamma_- = \Delta$, $\gamma'_+ = \Delta$, and $\gamma'_- = -\Delta$. From this the following claim follows.

[8] Precisely speaking, this relaxed goal should be specified by using some accuracy parameter $\varepsilon > 0$ for p' and r', which we will omit in the following explanation.

Claim 5. For any $i \in V_+ - \{1\}$, we have

$$\mathrm{E}[x_i] \;=\; \widetilde{\mathrm{E}}[x_i] - c^*(i) \left(\frac{2n(4\widetilde{\beta}^2)}{\alpha^2(2-\alpha)^2} \right) (\; \widetilde{\beta}\Delta - \Delta^2 \;).$$

Consider the case $\Delta < 0$; that is, we somehow guessed $\widetilde{\beta}$ smaller than β. In this case, the average margin $|\mathrm{E}[x_i]|$ is larger than $|\widetilde{\mathrm{E}}[x_i]|$, the value that we expect from our guessed $\widetilde{p}$ and $\widetilde{r}$. Note, however, the deviation of x_i from $\mathrm{E}[x_i]$ is also larger than what we expect, and desired confidence cannot be guaranteed if too small $\widetilde{\beta}$ is used.

Consider, on the other hand, the case $\widetilde{\beta} \geq \beta$ (i.e., $\Delta \geq 0$). For this case, it is easy to check that tolerance to the deviation is stronger. Therefore, for the theorem, it suffices to show that the average margin is large enough for our parameters. Now, as stated in Theorem 2, assume that $\beta \leq \widetilde{\beta} < (5/4)\beta$, which implies that $0 \leq \Delta \leq \beta/8 \leq \widetilde{\beta}/8$. Thus, for any $i \in V_+$ we have

$$\begin{aligned} \mathrm{E}[x_i] \;&=\; \widetilde{\mathrm{E}}[x_i] - \left(\frac{2n(4\widetilde{\beta}^2)}{\alpha^2(2-\alpha)^2} \right) (\; \widetilde{\beta}\Delta - \Delta^2 \;) \\ &\geq\; \frac{2n\widetilde{\beta}^4}{\alpha^2(2-\alpha)^2} - \frac{2n(4\widetilde{\beta}^3\Delta)}{\alpha^2(2-\alpha)^2} \;\geq\; \frac{2n\widetilde{\beta}^4}{\alpha^2(2-\alpha)^2} - \frac{2n(4\widetilde{\beta}^4/8)}{\alpha^2(2-\alpha)^2} \;\geq\; \widetilde{\mathrm{E}}[x_i]/2. \end{aligned}$$

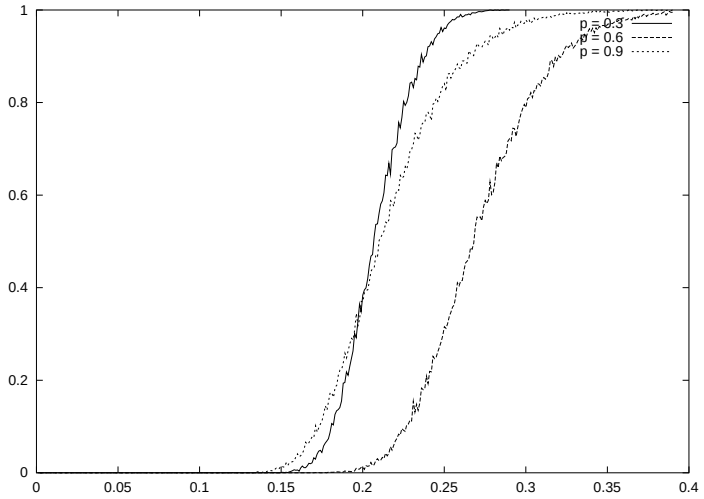
That is, the real average value is at least the half of the "pseudo average" computed from the parameters $\widetilde{p}$ and $\widetilde{r}$. Therefore, by using an argument similar to the one for Theorem 1, we can delive a similar condition for the value of the random variable x_i being positive. (Because the margin is now the half of what we expect from $\widetilde{p}$ and $\widetilde{r}$, we need roughly the half of the bounds in the proof of Theorem 1.) A similar condition can be derived for $i \in V_-$. Then by analyzing the probability that all these condition hold, we can show the desired bound for the success probability.

4 Some Experimental Results and Concluding Remarks

For some graph partitioning problems, we proposed a simple deterministic algorithm based on the belief propagation, and we investigated its performance and gave some theoretical justification. There are, however, many open questions on the performance and the behavior of our algorithm, which have been observed from computer experiments. Here we report on our computer experiments and ask related open questions.

For comparing the experiments of [DLP03], our experiments are conducted on *bipartite* graphs of size $n = 200$; vertices are divided into four groups U_+, V_+, U_-, and V_-, and edges are randomly generated with probability p between pairs (u, v) of vertices $u \in U_+$ and $v \in V_+$ (resp., $u \in U_-$ and $v \in V_-$) and with probability r between pairs (u, v) of vertices $u \in U_+$ and $v \in V_-$ (resp., $u \in U_-$ and $v \in V_+$). For three choices of p, $p = 0.3$, $p = 0.6$, and $p = 0.9$, we change $p - r$ from 0.001 to $\min(0.399, p)$ with 0.001 increment. For each choice of parameters, we generate 2,000 graphs and run `PartByPseudoBP` with sufficiently large MAXSTEP. (Recall that `PartByPseudoBP` is the same as `PseudoBP` of Figure 2 except that it outputs a classification instead of pseudo beliefs.)

Figure 3 shows the relation between $p - r$ and the *success probability*, the probability of obtaining a planted solution. Compared with the results reported in [DLP03], our algorithm



$p = 0.3$, $p = 0.6$, and $p = 0.9$

Figure 3: Success probability vs. $p - r$

seems to show better performance than the one proposed in [DLP03]. It is easy to see that, for each p (and n), there is some threshold and the success probability increases rapidly when $p-r$ gets larger than the threshold. When $p-r$ is fixed, among three choices of p, the performance of the algorithm seems the worst when $p = 0.6$. On the other hand, the success probability bound obtained by our theoretical analysis becomes lowest when $p = 1.0$. Of course, our theoretical bound is for the case MAXSTEP $= 2$, and the situation may differ when more iterations are allowed; but it is also likely that our bound is not sharp enough. One of the important open problems is to improve our analysis for obtaining a sharper bound and a bound for a few iterations.

The success probabilities are obtained by running the algorithm with sufficiently large MAXSTEP. Figure 4 shows the relation between $p - r$ and the number of iterations until all pseudo beliefs get stabilized. (Right one is obtained by overlapping Figure 3 onto the left one.)

Again it is easy to see that the algorithm terminates in at most 45 steps and that for sufficiently large $p - r$, the number of iterations is quite small, i.e., less than 10. Another important open problem is to give some upper bound for the number of iterations. We observe that pseudo beliefs get stabilized with values greater than the positive threshold th_+ or less than the negative threshold $-\mathrm{th}_+$. But the proportion of such overthreshold values becomes less than 1 and gets decreased when $p-r$ gets smaller than 0.15. Note that this is about the point when the number of iterations starts increasing. For very small $p - r$, the computed values stay in a very small range after the first iteration; thus, the algorithm (in our current implementation) regard this situation as a stable state, and it terminates in two iterations. But we do not know whether the values indeed get converged.

As shown in Figure 3, for small $p - r$, the probability that our algorithm does not yield a planted solution is large. But this does not necessarily mean that the algorithm makes an error. In fact, when $p - r$ is small, a planted solution is no longer the best for the Graph Bisection problem nor the Most Likely Partition problem. While it is hard to find the best solution, we

can check whether the output of the algorithm is better than the planted solution that is used to create the input graph. Figure 4 shows the relation between $p - r$ and the probability that the algorithm yields a partition whose likelihood is better than the one by the planted solution. It is challenging to verify (either experimentally or theoretically) that the output of the algorithm is almost optimal for the Molst Likely Partition problem.

Note that when the algorithm yields a solution different from the planted solution, it is often the case that the partition defined by the solution is not of equal size; that is, it is not a solution for the Graph Bisection problem.

References

[Bop87] R.B. Boppana, Eigenvalues and graph bisection: an average-case analysis, in *Proc. Symposium on Foundations of Computer Science*, 280-285, 1987.

[Betal87] T. Bui, S. Chaudhuri, F. Leighton, and M. Spiser, Graph bisection algorithms with good average behaviour, in *Combinatorica* 7, 171–191, 1987.

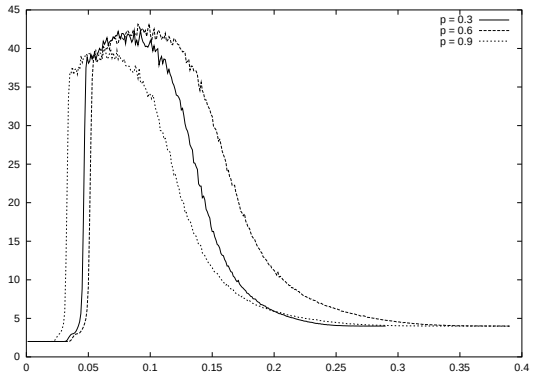
[CK01] A. Condon and R. Karp, Algorithms for graph partitioning on the planted partition model, *Random Str. and Algorithms* 18, 116–140, 2001.

[DLP03] D. Dubhashi, L. Laura, and A. Panconesi, Analysis and experimental evaluation of a simple algorithm for oollaborative filtering in planted partition models, in *Proc. FST TCS 2003*, 168–182, 2003.

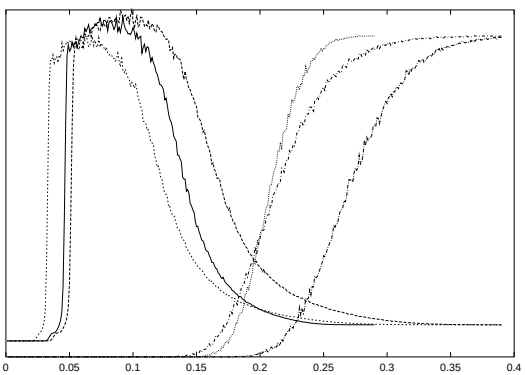
[DF89] M.E. Dyer and A.M. Frieze, The solution of some random NP-hard problems in polynomial expected time, *J. of Algorithms* 10, 451–489, 1989.

[GJ79] M.R. Garey, D.S. Johnson, *Computers and Intractability*, Bell Telephone Laboratories, Incorporated, 1979.

[GJS76] M. Garey, D. Johnson, and L. Stockmeyer, Some simplified NP-complete graph problems, in *Theoret. Comput. Sci.* 1, 237–267, 1976.

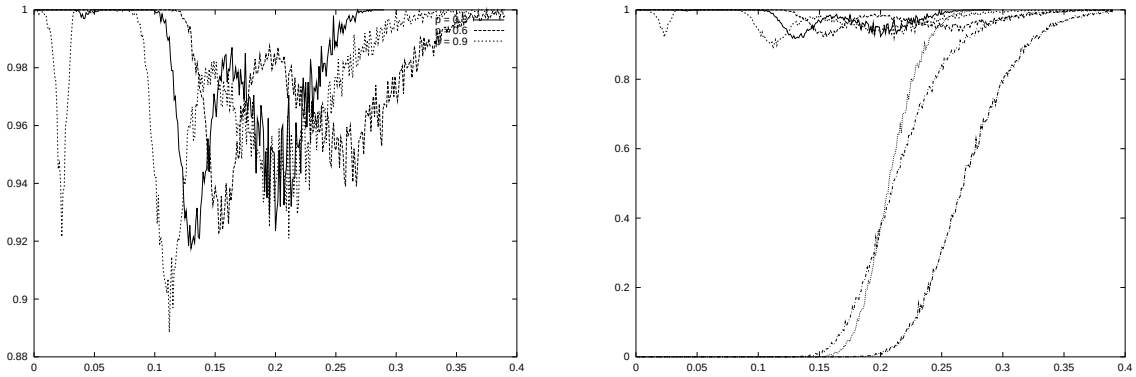


$p = 0.3$, $p = 0.6$, and $p = 0.9$



For comparison, Figure 3 is overlapped.

Figure 4: Average number of iterations vs. $p - r$



$p = 0.3$, $p = 0.6$, and $p = 0.9$ with graphs of Figure 3

Figure 5: Prob. of a better solution for MLP vs. $p - r$

[JS98] M. Jerrum and G. Sorkin, The Metropolis algorithm for graph bisection, *Discrete Appl. Math* 82(1-3), 155–175, 1998.

[HSS03] R. Hahnloser, H. Seung, and J. Slotine, Permitted and forbidden sets in threshold-linear networks, in *Neural Computation* 15, 621–638, 2003.

[1] R. Hahnloser, About the piecewise analysis of networks of linear threshold neurons, in *Neural Networks* 11, 691-697, 2002.

[MMC98] 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.

[Ons05] M. Onsjö, Master Thesis, 2005.

[Pea88] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers Inc., 1988.