Improved Algorithms for the Random Cluster Graph Model*

Ron Shamir† Dekel Tsur‡

Abstract

We model noisy clustering data using random graphs: Clusters correspond to disjoint sets of vertices. Two vertices from the same set (resp., different sets) share an edge with probability p (resp., $r < p$). We give algorithms that reconstruct the clusters from the graph with high probability. Compared to previous studies, our algorithms have lower time complexity and apply under wider parameter range.

1 Introduction

Clustering is a fundamental problem that has applications in many areas. We study the clustering problem using a random graph model: A *random cluster graph* is a random graph $G = (V, E)$ which is built by the following process: The vertex set V is a union of disjoint sets $V_1, \dots, V_m$ called *clusters*. Each two vertices from the same set are connected by an edge with probability p, and each two vertices from different sets are connected by an edge with probability $r < p$. The random choices are all independent. The *clustering problem* is, given a random cluster graph G, to find the clusters $V_1, \dots, V_m$.

The random cluster graph models a common situation in experimental data, where noise obscures the true clusters: The probability $1 - p$ is the *false negative* probability, i.e., the chance of incorrectly having no edge between two vertices of a cluster. Similarly, r is the *false positive* probability. If errors of each type are independent and identically distributed, one gets the above model.

In this paper we give several algorithms that solve the clustering problem with high probability. When addressing the clustering problem under the random graph model, several parameters are interrelated: Obviously, the smaller the gap $\Delta = p - r$, the harder the problem. Also, the size of the smallest cluster

*A preliminary version containing some of the results has been published in the Proceedings of the Eighth Scandinavian Workshop on Algorithm Theory (SWAT 02) [12].

†School of Computer Science, Tel Aviv University. E-Mail: rshamir@tau.ac.il

‡Dept. of Computer Science, Haifa University. E-Mail: dekelts@cs.haifa.ac.il

$k = \min_i |V_i|$ is limiting the performance, as very small clusters may be undetectable due to noise. The value k also bounds the number of clusters m. The challenge is to obtain provably good performance for a wide range of values for each parameter. All previous studies addressed the problem when the number of clusters is constant and most studied the case of two equal sized clusters. Our algorithms relax both of these assumptions simultaneously and at the same time achieve better running time.

Let us be more precise. Denote the total number of vertices in the graph by n. We give an $O(mn^2/\log n)$-time algorithm that solves the clustering problem with high probability, assuming that $k = \Omega(\Delta^{-1-\epsilon}\sqrt{n \log n})$ for some constant $\epsilon > 0$. The running time improves if k is asymptotically larger than its lower bound. Furthermore, if the sizes of the clusters are equal (or almost equal), we give an $O((m/\log n + 1)n^2)$-time algorithm that requires only $k = \Omega(\Delta^{-1}\sqrt{n}\log n)$ (here too, the running time improves if k is asymptotically larger).

The random graph model was studied by several authors [1–3, 5–9]. In these papers, the input is a random cluster graph, and the goal is to find a vertex partition which minimizes some function, e.g., the number of edges between different sets. It is not hard to show that w.h.p., the partition $V_1, \ldots, V_m$ is optimal, and therefore these problems are asymptotically equivalent to the clustering problem. A comparison between the results of these papers and our results is given in Table 1. The algorithms presented here have a wider range of provable performance than each of the previous algorithms, and are also faster when restricted to the same parameter range. For example, for unequal sized clusters, the algorithm of Ben-Dor et al. [1] requires $k = \Omega(n)$ and $\Delta = \Omega(1)$, while our algorithm can handle instances with $k = \Theta(\sqrt{n \log n})$, and instances with $\Delta = \Theta(n^{-1/2+\epsilon})$. Furthermore, under the requirements of Ben-Dor et al., the running time of our algorithm is $O(n \log n)$. For the case of two equal sized clusters, our algorithm handles almost the same range of Δ as the algorithm of Boppana [2], but our algorithm is faster, and is also more general since it handles as many as $m = \Theta(\sqrt{n}/\log n)$ clusters.

We note that Table 1 cites results as given in the papers, even though in several cases better results can be obtained by improving the analysis or by making small modifications to the algorithms. For example, the algorithm of Condon and Karp can be extended to the case when the number of clusters is non-constant. Also, the running time of the algorithm of Ben-Dor et al. can be improved to $O(n \log n)$ [14].

After obtaining our results, McSherry [10] gave a new polynomial algorithm for the clustering problem that requires that $k = \Omega(\Delta^{-1}\sqrt{n \log n} + \Delta^{-2/3}n^{2/3})$. For the case of equal sized clusters, our algorithm has a wider range of k when $\Delta = w(n^{-1/2}\log^3 n)$, while McSherry's algorithm has a wider range of k when $\Delta = o(n^{-1/2}\log^3 n)$ (note that McSherry's algorithm has a lower bound of $\Omega(n^{-1/2}\sqrt{\log n})$ on Δ). For the case of unequal clusters, our algorithm has a wider range of k

Paper	Requirements		Complexity
	k	Δ	
Ben-Dor et al. [1]	$\Omega(n)$	$\Omega(1)$	$O(n^2 \log^{O(1)} n)$
This paper	$\Omega(\Delta^{-1-\epsilon}\sqrt{n \log n})$	$\Omega(n^{-1/2+\epsilon})^*$	$O(mn^2/\log n)$

(a) General case.

Paper	Requirements		Complexity
	m	Δ	
Dyer & Frieze [6]	2	$\Omega(n^{-1/4} \log^{1/4} n)$	$O(n^2)$
Boppana [2]	2	$\Omega(\sqrt{p}n^{-1/2}\sqrt{\log n})$	$n^{O(1)}$
Jerrum & Sorkin [8]	2	$\Omega(n^{-1/6+\epsilon})$	$O(n^3)$
Jules [9]	2	$\Omega(1)$	$O(n^3)$
Condon & Karp [5]	$O(1)$	$\Omega(n^{-1/2+\epsilon})$	$O(n^2)$
Carson & Impagliazzo [3]	2	$\omega(\sqrt{p}n^{-1/2} \log n)$	$O(n^2)$
Feige & Kilian [7]	2	$\Omega(\sqrt{p}n^{-1/2}\sqrt{\log n})$	$n^{O(1)}$
This paper	$O(\sqrt{n}/\log n)^*$	$\Omega(mn^{-1/2} \log n)$	$O((\frac{m}{\log n} + 1)n^2)$

(b) Equal sized clusters.

Table 1: Results on the clustering problem (sorted in chronological order). For the comparison, the lower bound $k = \Omega(\Delta^{-1}\sqrt{n} \log n)$ of our algorithm for equal sized clusters was translated to a lower bound on Δ using the fact that $k \leq n/m$. Note that all previous papers assume $m = 2$ or $m = O(1)$ (the requirement $k = \Omega(n)$ in [1] implies that $m = O(1)$), and all except [1] assume equal sized clusters. For the values that are marked by $*$, no implicit requirement is made, and the requirement is implied by the bound on the other parameter.

when $\Delta \geq n^{-1/2+\epsilon}$ for some $\epsilon > 0$, while McSherry's algorithm has a wider range of k when Δ is smaller.

1.1 Outline of our approach

In the rest of this section we outline the basic techniques that we use and their differences from previous studies. Our approach is similar to that of the algorithm of Condon and Karp [5] (we will refer to it as the CK algorithm). For a vertex v in a graph G, and a set S of vertices of G, let $d_S(v)$ denote the number of neighbors of v in S. We will first give a detailed description of the CK algorithm (in a slightly simplified version, for clarity):

1. Begin with empty sets L and R. Choose $n/4$ pairs of vertices (without repetitions). Repeatedly, for each pair v, w, compute the expressions

$d_L(v)-d_R(v)$ and $d_L(w)-d_R(w)$. Add the vertex whose expression is larger to L, and add the other vertex to R.

2. Sort all the vertices in decreasing order according to their $d_L(\cdot)$ values, and find the largest gap in the sorted sequence. If this gap is small, output V and stop. Otherwise, put the vertices before the gap into a set L', and the vertices after the gap into a set R'.

3. Recursively solve the problem on the subgraph induced by L' and on the subgraph induced by R'.

The analysis of the algorithm is based on the notion of imbalance: The L, R-*imbalance* of a cluster V_i is the number of vertices of V_i in L minus the number of vertices of V_i in R. The *imbalance of* L, R is the maximum value amongst the L, R-imbalance of the clusters. Let $l_1, \ldots, l_m$ denote the L, R-imbalances of the cluster $V_1, \ldots, V_m$ at some point of the algorithm, and suppose w.l.o.g. that $l_1 > l_2 \geq \cdots \geq l_m$. Condon and Karp showed that during the first step of the algorithm, the imbalance of L, R behaves like a random walk with growing bias for increase. The general idea of the proof is that the imbalance changes only when handling a pair v, w in which one vertex is from V_1 and the other vertex is not. Furthermore, for such a pair, it is more likely that the vertex from V_1 will be added to L, and therefore the imbalance will increase. More precisely, given some vertices $v, w \notin L \cup R$, the random variable $X = (d_L(v)-d_R(v))-(d_L(w)-d_R(w))$ is a sum of $4|L|$ independent random variables ($2|L|$ variables are Bernoulli variables and the rest are the negated Bernoulli variables). The expectation of X depends on the L, R-imbalances of the clusters that contain v and w: If $v \in V_i$ and $w \in V_j$ then the expectation of X is $\Delta|L|\cdot(l_i-l_j)$. Using estimates on sums of independent random variables (Esseen's inequality), it is shown that the probability that v will be added to L (which is equal to the probability that $X > 0$ plus half of the probability that $X = 0$) is $1/2 + \Omega(\min(\frac{(l_i-l_j)\Delta}{\sqrt{|L|}}, 1))$ for $i > j$. It follows that with high probability (w.h.p.), at the end of Step 1, the imbalance of L, R is $\Theta(n)$.

We now consider the second step of the algorithm, which we will call the *splitting step*, and suppose that there are at least two clusters. Using Chernoff bounds, w.h.p., for every vertex v, the deviation of $d_L(v)$ from its expectation is at most $\alpha = n^{1/2+\epsilon/2}$. The expectation of $d_L(v)$ depends on the imbalance of the cluster that contains v: If $v \in V_i$ then the expectation of $d_L(v)$ is roughly $A_i = \Delta(\frac{n}{4m}+l_i/2)+r\frac{n}{4}$. There is an index i such that $l_i-l_{i+1} \geq (l_1-l_m)/(m-1) = \Omega(n/m)$ (the last equality follows from the fact that $l_1 = \Theta(n)$ and $l_m \leq 0$). Therefore, $A_i - A_{i+1} = \Omega(\Delta n/m) = \Omega(n^{1/2+\epsilon})$, and the largest gap in the sorted sequence of $d_L(\cdot)$ values is at least $(A_i - A_{i+1}) - 2\alpha = \Omega(n^{1/2+\epsilon})$. It follows that w.h.p., every cluster V_i, is contained either in L' or in R'.

We now give a short description of our algorithms. The main difference between our algorithm and the CK algorithm is the distribution of the l_i-s. The success of the splitting step in the CK algorithm depends on having a large gap in the sequence of the l_i-s. As shown previously, this gap is $\Omega(n/m)$, and this bound is true also in case m is not a constant (this case was not analyzed by Condon and Karp). Furthermore, this lower bound is likely to be tight: Condon and Karp conjectured (and showed in simulations) that w.h.p., at the end of Step 1, the l_i-s are distributed uniformly between $l = (1 - 1/m)\frac{n}{4m}$ and $-l$, namely $l_1, \ldots, l_m \approx l, (1 - \frac{2}{m})l, (1 - \frac{4}{m})l, \ldots, -(1 - \frac{2}{m})l, -l$. This implies that the largest gap in the l_i-s sequence is $\Theta(n/m)$. In contrast, our algorithms were designed in order to keep the values of $l_2, \ldots, l_m$ much smaller than l_1, (namely, $l_2, \ldots, l_m \leq (1 - \delta)l_1$ for some constant $\delta > 0$), and use this fact to achieve a $\Theta(n)$ gap. This allows us to give better algorithms for the case where m is non-constant. Furthermore, since it is known that there is a large gap between l_1 and l_2, we can use a modified splitting step: Instead of finding the maximum gap in the sorted $d_L(\cdot)$ values, we find the first gap which is large enough. The result is that w.h.p. the set L' will contain only one cluster (V_1).

In order to build the sets L and R with the desired imbalances, we build a series of pairs of sets L_i, R_i, where the pair L_i, R_i is built from L_{i-1}, R_{i-1} using a process similar to the CK algorithm: We choose disjoint pairs of vertices, and for each pair v, w we compute the expressions $d_{L_{i-1}}(v) - d_{R_{i-1}}(v)$ and $d_{L_{i-1}}(w) - d_{R_{i-1}}(w)$. The vertex whose expression is larger is added to L_i, and the other vertex to R_i. (The pair L_0, R_0 is built using a similar process). Note that unlike the CK algorithm, the sets L_{i-1}, R_{i-1}, L_i, and R_i are disjoint, so we have independence between the pairs of vertices, which simplifies the analysis of the algorithm. We show that the imbalances of the pairs L_i, R_i increases at each iteration.

We note that the algorithms presented here might not be directly applicable in real-life applications since our analysis holds only for large n, and since real life data might not behave like the random cluster graph model. However, the ideas we use might be useful for building good heuristic algorithms. For example, Ben-Dor et al. gave a heuristic algorithm (called CAST) based on their theoretical algorithm for the random cluster graph model, and demonstrated that the heuristic performs well in real clustering applications [1].

The rest of this paper is organized as follows: Section 2 contains notation and estimates on the sum of independent random variables. Section 3 and Section 4 contain the basic ideas for the algorithms. The algorithm for the case of almost equal sized clusters is given in Section 5, and the algorithm for the general case is given in Section 6. We provide at the end of the paper a list of the main symbols used in the paper and their meaning.

2 Preliminaries

For a graph $G = (V, E)$, and a vertex $v \in V$, we denote by $N(v)$ the set of neighbors of v. We use $d(v)$ to denote the degree of a vertex v, namely $d(v) = |N(v)|$. For a vertex v, and a set S of vertices, denote $d_S(v) = |N(v) \cap S|$. In particular, for $u \neq v$, $d_{\{u\}}(v)$ is equal to 1 if $(u, v) \in E$ and 0 otherwise. For a set of vertices S, let G_S denote the subgraph of G induced by S.

We use the O-notations O, Ω and o with their usual meaning. However, we write $f \leq O(g)$ instead of the more common $f = O(g)$ to emphasize that an upper bound on f is given. We also use $f \leq \hat{o}(g)$ to denote that $f(n) \leq c \cdot g(n)$ for some constant c that can be made arbitrarily small, depending on other constants (This implies in particular that $f \leq O(g)$). For example, if we have the requirement $k \geq \Omega(\sqrt{n})$, then we can write that $m \leq n/k \leq \hat{o}(\sqrt{n})$ since the hidden constant in the latter inequality can be made arbitrarily small by increasing the hidden constant in the former inequality.

For proving the correctness of our algorithms we use the following theorems which give estimates on the sum of independent random variables. The following theorem is derived from Chernoff [4].

Theorem 2.1. *Let $X_1, \ldots, X_n$ be independent Bernoulli random variables, and let $X = \sum_{i=1}^{n} X_i$. Then,* $\mathrm{P}\left[|X - \mathrm{E}[X]| \geq a\sqrt{3\mathrm{E}[X]}\right] \leq 2e^{-a^2}$ *for every $a \geq 0$.*

Theorem 2.2 (Esseen's Inequality). *Let $X_1, \ldots, X_n$ be independent random variables such that $\mathrm{E}[X_i] = 0$ and $\mathrm{E}[|X_i|^3] < \infty$, for $i = 1, \ldots, n$. Let $B_n = \sum_{i=1}^{n} \mathrm{E}[X_i^2]$ and $L_n = B_n^{-3/2} \sum_{i=1}^{n} \mathrm{E}[|X_i|^3]$. Then $\left|\mathrm{P}\left[B_n^{-1/2} \sum_{i=1}^{n} X_i < x\right] - \Phi(x)\right| \leq AL_n$ where A is an absolute constant and $\Phi(x)$ denotes the normal $(0, 1)$ cumulative distribution function.*

For a proof of Esseen's inequality see [11, p. 111]. Esseen's inequality holds for $A = 0.8$ [13].

Corollary 2.3. *Let $X_1, \ldots, X_n$ be independent random variables such that $\mathrm{E}[|X_i|^3] < \infty$, for $i = 1, \ldots, n$. Let $B_n = \sum_{i=1}^{n} \mathrm{Var}(X_i)$, $L_n = B_n^{-3/2} \sum_{i=1}^{n} \mathrm{E}[|X_i - \mathrm{E}[X_i]|^3]$, and $X = \sum_{i=1}^{n} X_i$. Then $|\mathrm{P}[X > 0] - \Phi(\mathrm{E}[X]/\sqrt{B_n})| \leq 0.8L_n$ and $|\mathrm{P}[X < 0] - (1 - \Phi(\mathrm{E}[X]/\sqrt{B_n}))| \leq 0.8L_n$.*

Proof. For the first part of the corollary, define $Y_i = \mathrm{E}[X_i] - X_i$ and apply Theorem 2.2 on $Y_1, \ldots, Y_n$ with $x = \mathrm{E}[X]/\sqrt{B_n}$. The second part is proved by taking $Y_i = X_i - \mathrm{E}[X_i]$ and $x = -\mathrm{E}[X]/\sqrt{B_n}$. ■

We also use the following lemma.

Lemma 2.4. *Let A be a set with n elements, and B be a subset of A with k elements. Let S' be a random subset of A, and let S be a random subset of $A - S'$ of size s. Then* $\mathrm{P}\left[||B \cap S| - \frac{k}{n}s| \geq a\sqrt{3(k/n)s}\right] \leq 6\sqrt{k}e^{-a^2}$ *for every $a \geq 0$.*

Proof. Let $X_1, \ldots, X_n$ be independent random variables which have Bernoulli distribution with parameter k/n. Let $X = \sum_{i=|S'|+1}^{|S'|+s} X_i$ and $Y = \sum_{i=1}^{n} X_i$. Fix some $a \geq 0$, and let Z denote the event that $|X - \frac{k}{n}s| \geq a\sqrt{3(k/n)s}$. We have that

$$\mathrm{P}\left[||B \cap S| - \frac{k}{n}s| \geq a\sqrt{3\frac{k}{n}s}\right] = \mathrm{P}\left[Z|Y=k\right] = \frac{\mathrm{P}\left[Z \cap (Y=k)\right]}{\mathrm{P}\left[Y=k\right]} \leq \frac{\mathrm{P}\left[Z\right]}{\mathrm{P}\left[Y=k\right]}.$$

By Theorem 2.1, $\mathrm{P}\left[Z\right] \leq 2e^{-a^2}$. Furthermore,

$$\mathrm{P}\left[Y=k\right] = \binom{n}{k}\left(\frac{k}{n}\right)^k\left(1-\frac{k}{n}\right)^{n-k} = \frac{n!}{k!(n-k)!} \cdot \frac{k^k(n-k)^{n-k}}{n^n}.$$

Stirling formula states that $\sqrt{2\pi l}\left(\frac{l}{e}\right)^l \leq l! \leq 1.09\sqrt{2\pi l}\left(\frac{l}{e}\right)^l$. Hence,

$$\begin{aligned}\mathrm{P}\left[Y=k\right] &\geq \frac{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n}{1.09\sqrt{2\pi k}\left(\frac{k}{e}\right)^k \cdot 1.09\sqrt{2\pi(n-k)}\left(\frac{n-k}{e}\right)^{n-k}} \cdot \frac{k^k(n-k)^{n-k}}{n^n} \\ &= \frac{1}{1.09^2\sqrt{2\pi}}\sqrt{\frac{n}{k(n-k)}} \geq \frac{1}{3\sqrt{k}}.\end{aligned}$$

The lemma follows from the above bounds. ■

In the following, we say that an event happens with high probability (w.h.p.) if its probability is $1 - n^{-\Omega(1)}$. Using this terminology we obtain the following corollaries of Theorem 2.1 and Lemma 2.4, respectively.

Corollary 2.5. *Under the conditions of Theorem 2.1, w.h.p.,* $|X - \mathrm{E}[X]| \leq O(\sqrt{\mathrm{E}[X]\log n})$.

Corollary 2.6. *Under the conditions of Lemma 2.4, w.h.p.,* $||B \cap S| - \frac{k}{n}s| \leq O(\sqrt{(k/n)s\log n})$.

We also obtain the following corollary:

Corollary 2.7. *Under the conditions of Lemma 2.4, and if additionally* $S' = \phi$ *and* $s \leq n/2$, *then w.h.p.,* $|\frac{|B-S|}{n-s} - \frac{k}{n}| \leq O(\sqrt{(k/n)n^{-1}\log n})$.

Proof. Clearly,

$$\left|\frac{|B-S|}{n-s} - \frac{k}{n}\right| = \frac{\left||B-S| - (1-\frac{s}{n})k\right|}{n-s} = \frac{\left||B-S| - k + \frac{s}{n}k\right|}{n-s} = \frac{\left||B \cap S| - \frac{s}{n}k\right|}{n-s},$$

and by Corollary 2.6, w.h.p.,

$$\left|\frac{|B-S|}{n-k} - \frac{s}{n}\right| \leq O(\frac{1}{n-s}\sqrt{(k/n)s\log n}) \leq O(\sqrt{(k/n)n^{-1}\log n}).$$ ■

3 The basic algorithm

In this section we give a top-level description of our algorithms.

Let $G = (V, E)$ be a random cluster graph. Denote $A_i = |V_i|$, $a_i = |V_i|/n$, and $\Gamma = \max_i |\frac{|V_i|}{n} - \frac{1}{m}|$. A set $S \subseteq V$ is called a *subcluster* if $S \subseteq V_i$ for some cluster V_i. An induced subgraph G_S is called a *cluster collection* if for all i, either $V_i \subseteq S$ or $V_i \subseteq V - S$. Suppose we have a procedure $\mathrm{Find}(G, V')$ that receives a random cluster graph $G = (V, E)$ and a set $V' \subseteq V$, and returns a subcluster of size $\Omega(\log n/\Delta^2)$, by only considering vertices and edges in $G_{V'}$. Then, we use the following algorithm for solving the clustering problem: Repeatedly, find a subcluster S in the graph, find the cluster that contains S (using a procedure similar to the splitting step of the CK algorithm), and remove the cluster from the graph. Formally, the algorithm is as follows:

Solve(G)

1. Randomly partition the vertices of G into equal sized sets W_1, W_2, and W_3.
2. $S_0 \leftarrow \mathrm{Find}(G, W_1)$.
3. Let S be a random subset of S_0 of size $s = \Theta(\log n/\Delta^2)$. Let $v_1, \dots, v_{n/3}$ be an ordering of W_2 such that $d_S(v_1) \geq d_S(v_2) \geq \cdots \geq d_S(v_{n/3})$. Let $D = \Theta(\sqrt{s \log n})$. If $\max_j\{d_S(v_j) - d_S(v_{j+1})\} < D$, output V and stop.
4. Let j be an index such that $d_S(v_j) - d_S(v_{j+1})$ is maximum, and let $S_2 = \{v_1, \dots, v_j\}$.
5. Let S' be a random subset of S_2 of size s. Let $w_1, \dots, w_{n/3}$ be an ordering of W_3 such that $d_{S'}(w_1) \geq \cdots \geq d_{S'}(w_{n/3})$. Let j' be an index such that $d_{S'}(w_{j'}) - d_{S'}(w_{j'+1})$ is maximum, and let $S_3 = \{w_1, \dots, w_{j'}\}$.
6. Let S'' be a random subset of S_3 of size s. Let $u_1, \dots, u_{n/3}$ be an ordering of W_1 such that $d_{S''}(u_1) \geq \cdots \geq d_{S''}(u_{n/3})$. Let j'' be an index such that $d_{S''}(u_{j''}) - d_{S''}(u_{j''+1})$ is maximum, and let $S_1 = \{u_1, \dots, u_{j''}\}$.
7. Output $S_1 \cup S_2 \cup S_3$ and delete the vertices in $S_1 \cup S_2 \cup S_3$ from G.
8. Goto 1.

See also Figure 1. Here and in the following, n is the number of vertices in the current subgraph of the input graph G. For simplicity, we assume that n is divisible by 3. The description of the algorithm above (and the rest of the algorithms in this paper) is slightly incomplete since the hidden constants are not specified. We chose not to write the constants explicitly as it would have affected the readability of the proofs.

Note that the partition of the graph into three parts is necessary in order to avoid dependencies: This partition guarantees, for example, that on step 3 of the algorithm, for every $u \in S$ and $v \in W_2$, the existence of the an edge between

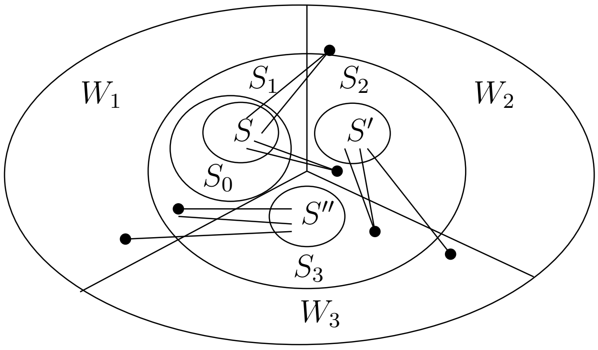

Figure 1: The sets created by algorithm Solve. The edges represented are counted by the algorithm. For example, the set S_2 is built by looking at the edges between W_2 and S.

u and v is independent of the choice of the set S. To see this, suppose that the edges of the graph are not chosen before the algorithm begins, but instead, the existence of an edge is determined randomly when the algorithm first checks the existence of that edge. Since procedure Find only considers edges with both endpoints in W_1, the existence of the edges between W_1 and W_2 has not been determined when the algorithm enters step 3, so for each $u \in W_1$ and $v \in W_2$, the event that there is an edge between u and v is independent of the choice of S. In fact, this would remain true, even if the set S is chosen by an adversary that can only look at edges in G_{W_1}. Note that partitioning the graph into two sets and using the same arguments as above would not be enough, since some edges between the two parts would be considered twice by the corresponding algorithm. However, it is possible to show that the algorithm is correct in this case, using additional arguments.

Let $T(G)$ denote the running time of procedure Find on an input graph G.

Lemma 3.1. *If procedure* Find *returns w.h.p. a subcluster of size* $\Omega(\log n/\Delta^2)$, *then w.h.p., algorithm* Solve *solves the clustering problem. W.h.p., the running time of algorithm* Solve *is* $O(mT(G) + mn \log n/\Delta^2)$.

Proof. We shall prove that the failure rate in one iterations is n^{-c} for some constant c. The probability of failure of the algorithm is at most m times the probability of failure in one iteration, and since $m \leq n$ and c can be chosen sufficiently large, we obtain that the overall failure probability is $n^{-\Omega(1)}$. We shall show that if G consists of one cluster, then w.h.p. the algorithm stops at Step 3, and otherwise, if $S_0 \subseteq V_i$ then w.h.p., $S_1 \cup S_2 \cup S_3 = V_i$.

W.l.o.g. we assume that $S_0 \subseteq V_1$. Since the edges between W_1 and W_2 are independent of the choice of S, by Corollary 2.5 we have that for a vertex $v \in W_2$, the probability that $|d_S(v) - \mathrm{E}[d_S(v)]| \geq \frac{1}{2}D$ is $n^{-c'}$ for a constant c' that depends on D. The probability that this happens for at least one vertex in W_2 is therefore at most $\frac{n}{3} \cdot n^{-c'}$. Thus, by choosing the constant in D in way that makes c'

sufficiently large, we obtain that w.h.p., $|d_S(v) - \mathrm{E}[d_S(v)]| < \frac{1}{2}D$ for all $v \in W_2$. For $v \in V_1$ we have that $\mathrm{E}[d_S(v)] = sp$, and for $v \notin V_1$ we have $\mathrm{E}[d_S(v)] = sr$. Hence, if G consists of one cluster, then w.h.p. $|d_S(v) - d_S(v')| < D$ for all $v, v' \in W_2$, and therefore, algorithm Solve stops at Step 3. Otherwise, for two vertices $v, v' \in W_2$, if either $v, v' \in V_1$ or $v, v' \notin V_1$ then w.h.p. $|d_S(v) - d_S(v')| < D$, and if $v \in V_1$ and $v' \notin V_1$ then w.h.p. $d_S(v) - d_S(v') > s\Delta - D \geq D$ where the last inequality is achieved by choosing appropriate constants for s, D so $s\Delta \geq 2D$. It follows that algorithm Solve does not stop at Step 3, and furthermore, $S_2 = V_1 \cap W_2$. Using similar arguments we show that w.h.p. $S_3 = V_1 \cap W_3$ and $S_1 = V_1 \cap W_1$.

From the correctness of the algorithm, we have that w.h.p. algorithm Solve performs exactly m iterations. The time complexity of a single iteration is $O(T(G) + n \log n / \Delta^2)$, where the second term is the time for computing the degrees at steps 3, 5, and 6. (Sorting the degrees can be done in $O(n)$ time by bin sorting.) ■

Note that algorithm Solve requires the knowledge of Δ. However, if instead of Δ we use some known lower bound Δ' on Δ, then the algorithm remains correct, and the running time becomes $O(mT(G) + mn \log n / \Delta'^2)$.

If the value of $mT(G) + mn \log n / \Delta^2$ is known (or if an upper bound on this value is known), then the algorithm can be modified to to stop after $O(mT(G) + mn \log n / \Delta^2)$ steps with a "Failure" output. This change does not affect the correctness of the algorithm, and the running time is always $O(mT(G) + mn \log n / \Delta^2)$.

3.1 Handling large clusters and close edge probabilities

In order to build the procedure Find, we will need the following additional requirements on the input graph:

(R1) $k \geq \Omega(\sqrt{n} \log^\alpha n / \Delta^\beta)$ for some $\alpha, \beta > 0$.

(R2) $p, r \in [\frac{1}{4}, \frac{3}{4}]$.

(R3) $\max_i a_i < \frac{2}{3}$.

While requirement of the type (R1) is understandable, we would like to get rid of (R2) and (R3). We will show how to deal with these requirements in the rest of this section.

To eliminate (R2), we transform the input graph to a graph that always satisfies the requirement using the following algorithm:

Solve$_2(G)$:

1. Build a graph G' on the vertex set of G in the following way: For every pair of vertices u, v, add the edge (u, v) to G' with probability $\frac{3}{4}$ if (u, v) is an edge in G, and with probability $\frac{1}{4}$ otherwise.

2. Run Solve(G').

Lemma 3.2. *If procedure* Find *returns w.h.p. a subcluster of size* $\Omega(\log n/\Delta^2)$ *on graphs that satisfy (R1)–(R3), then w.h.p., algorithm* Solve_2 *solves the clustering problem on graphs that satisfy (R1) and (R3). The running time of algorithm* Solve_2 *is* $O(mT(G') + mn \log n/\Delta^2)$.

Proof. Clearly, G' is a random cluster graph on the clusters $V_1, \dots, V_m$ with edge probabilities $p' = \frac{3}{4}p + \frac{1}{4}(1-p) = \frac{1}{2}p + \frac{1}{4}$ and $r' = \frac{1}{2}r + \frac{1}{4}$. G' satisfies (R1) and furthermore, every cluster collection G'' of G' satisfies (R1) as the size of the smallest cluster in G'' is greater or equal to the size of the smallest cluster in G. As $p', r' \in [\frac{1}{4}, \frac{3}{4}]$ and $p' - r' = \Theta(\Delta)$, by Lemma 3.1 Solve(G') returns w.h.p. the correct partition. ∎

To eliminate (R3) we use a procedure that recursively partitions the input graph into cluster collections until subgraphs that satisfy (R3) are obtained. This is done by the following algorithm:

$\text{Solve}_3(G)$:

1. Let $S = \phi$. Independently add each vertex $v \in V$ to S with probability $q = \Theta(\log n/\Delta^2 n)$. If $|S| \geq 2qn$, output 'Failure' and stop.
2. Let $v_1, \dots, v_n$ be an ordering of V such that $d_S(v_1) \geq d_S(v_2) \geq \cdots \geq d_S(v_n)$. Let $D = \Theta(\sqrt{qn \log n})$. If $\max_j\{d_S(v_j) - d_S(v_{j+1})\} < D$ then call $\text{Solve}_2(G)$ and stop.
3. Let j be an index such that $d_S(v_j) - d_S(v_{j+1})$ is maximum, and let $R = \{v_1, \dots, v_j\}$. Call $\text{Solve}_3(G_R)$ and $\text{Solve}_3(G_{V-R})$.

For the following lemma we assume that T satisfies $\sum_{i=1}^{l} T(G_i) \leq T(G)$ for any partition of G into vertex disjoint cluster collections $G_1, \dots, G_l$, and furthermore, for the graph G' that is built by $\text{Solve}_2(G)$, $T(G') \leq O(T(G))$. These assumptions are satisfied by the time complexity function of the procedure we give in Section 6.

Lemma 3.3. *If procedure* Find *returns w.h.p. a subcluster of size* $\Omega(\log n/\Delta^2)$ *on graphs that satisfy (R1)–(R3) then w.h.p., algorithm* Solve_3 *solves the clustering problem on graphs that satisfy (R1). The running time of algorithm* Solve_3 *is* $O(mT(G) + mn \log n/\Delta^2)$.

Proof. First, by Corollary 2.5, w.h.p. $|S| \leq qn + O(\sqrt{qn \log n}) \leq 2qn$, so algorithm Solve_3 does not stop at Step 1.

Consider the first iteration of the algorithm. Suppose that $a_1 \geq 2/3$, so $a_i \leq 1/3$ for all $i > 1$. By Corollary 2.5, w.h.p. $|d_S(v) - \mathrm{E}[d_S(v)]| < \frac{1}{2}D$ for all $v \in V$. For $v \in V_i$ we have $\mathrm{E}[d_S(v)] = (A_i - 1)qp + (n - A_i)qr = A_i q\Delta - qp + nqr$, so for $v \in V_1$ and $v' \in V_i$ (for $i \neq 1$) we have $\mathrm{E}[d_S(v)] - \mathrm{E}[d_S(v')] = (A_1 - A_i)q\Delta \geq \frac{1}{3}nq\Delta$.

Therefore, w.h.p. $\min_{v \in V_1} d_S(v) - \max_{v \in V-V_1} d_S(v) > \frac{1}{3}nq\Delta - D \geq D$ where the inequality follows by choosing the constants in q and D appropriately.

Hence, if at Step 2, $\max_j\{d_S(v_j) - d_S(v_{j+1})\} < D$ then w.h.p. $\max_i a_i < 2/3$. Otherwise, w.h.p. G_R and G_{V-R} are cluster collections (as for two vertices v, v' from the same cluster we have w.h.p. $|d_S(v) - d_S(v')| < D$). Let $G_1, \ldots, G_r$ denote the graphs on which algorithm Solve$_2$ is called during the run of Solve$_3(G)$. From the above, we have that w.h.p. $G_1, \ldots, G_r$ are cluster collections, and each G_i satisfies (R3). Furthermore, each G_i satisfies (R1). Therefore, the correctness of Solve$_3$ follows from Lemma 3.2.

We now compute the running time of algorithm Solve$_3$. We have $r \leq m$ as each G_i is nonempty. Therefore, the total time not including the calls to Solve$_2$ is $O(r \cdot qn^2) = O(mn \log n/\Delta^2)$. Let $n(G)$ denote the number of vertices in a graph G. The total time of the calls to Solve$_2$ is $O(\sum_{i=1}^{r}(mT(G_i) + mn(G_i)\log n/\Delta^2)) = O(mT(G) + mn\log n/\Delta^2)$. ■

4 The partition procedure

In the previous section, we presented the basic structure of our algorithms. However, we still need to show how to build procedure Find, namely, how to find a subcluster of size $\Omega(\log n/\Delta^2)$. In this section we give the main ideas, and we will use these ideas in sections 5 and 6 to explicitly build procedure Find.

One of the key elements we use is the notion of imbalance which was also used in [8] and [5]. We use here a slightly different definition for imbalance than the one used in these papers (and in the introduction): For two disjoint sets L, R of vertices of equal size, we define the *L, R-imbalance of V_i*, denoted $\mathrm{I}(V_i, L, R)$, by

$$\mathrm{I}(V_i, L, R) = \frac{|V_i \cap L| - |V_i \cap R|}{|L|}.$$

The *imbalance of* L, R is the maximum value amongst $\mathrm{I}(V_1, L, R), \ldots, \mathrm{I}(V_m, L, R)$, and the *secondary imbalance of* L, R is the second largest value (the secondary imbalance is equal to the imbalance if the maximum value appears more than once). We will show that given two sets L, R with "large" imbalance, and "small" secondary imbalance, it is possible to generate a large subcluster. Therefore, our goal is to generate the sets L, R with such properties.

Let T be a subset of V. Let $f\colon T \to \mathbb{N}$ be some function that depends on the edges of G. Since G is a random graph, we have that each $f(v)$ is a random variable. The function f is called a *cluster function* if $\{f(v)|v \in T\}$ are independent random variables, and for every $u, v \in T$ that belong to the same cluster, $f(u), f(v)$ have the same distribution. For example, $f(v) = d_{V-T}(v)$ is a cluster function.

If the values of f for vertices of one cluster are always greater than the values of f for the vertices of the other clusters, then we can easily generate a subcluster by picking $\Theta(\log n/\Delta^2)$ vertices with largest f-value. However, we are able to give such a cluster function f only when Δ is large. For smaller value of Δ, we are able to give a cluster function f with the following property: The expectation of $f(v)$ for vertices of one cluster, say V_1, is larger than the expectation of $f(v)$ for vertices in the other clusters, and the variance of $f(v)$ is "small" for all v. We will use this property of f in the following procedure (here T is a random set of vertices):

Partition(G, T, f):

1. Begin with two empty sets L, R. Randomly partition the vertices of T into pairs. For each pair v, w, place the vertex with larger f-value into L and the other into R, breaking ties randomly.
2. Return L, R.

We note that procedure Partition is very similar to the algorithm of Condon and Karp [5], and the cluster function of Lemma 4.3 is also used in [5].

Let $p_{ij}^{>}(f) = \mathrm{P}\left[f(v) > f(w) | v \in V_i, w \in V_j\right]$ (we will write $p_{ij}^{>}$ if f is clear from the context), and let $p_{ij}^{=}(f)$ and $p_{ij}^{<}(f)$ be the conditional probabilities that $f(v) = f(w)$ and $f(v) < f(w)$, respectively. Let $c_i(f) = 2a_i \sum_{j \neq i} a_j (p_{ij}^{>}(f) - p_{ij}^{<}(f))$. Suppose that L, R are the output of Partition(G, T, f) and denote $b_i = \mathrm{I}(V_i, L, R)$.

Theorem 4.1. *If T is a random set of size $2l \leq n/2$, $k \geq \Omega(\log n)$, and $\max_i a_i \leq 2/3$, then w.h.p. $|b_i - c_i(f)| \leq O(\sqrt{a_i l^{-1} \log n}(1 + \sqrt{a_i m}))$ for all i.*

Proof. Since T is a random set, the process of choosing T and then partitioning T into pairs, is equivalent to the process of selecting vertices $u^1, \ldots, u^{2l}$ from V (one after another) and pairing u^{2t-1} and u^{2t} for each t. Denote $a_j^r = \mathrm{P}\left[u^r \in V_j\right]$. Clearly, $a_j^r = |V_j - \{u^1, \ldots, u^{r-1}\}| / |V - \{u^1, \ldots, u^{r-1}\}|$. By Corollary 2.7, we have w.h.p. that $|a_j - a_j^r| \leq O(\sqrt{a_j n^{-1} \log n})$ for all j and r. In the following we assume that these inequalities hold.

We denote by I_i^1 (resp., I_i^2) the number of indices t for which $u^{2t-1} \in V_i$, $u^{2t} \notin V_i$, and u^{2t-1} (resp., u^{2t}) is inserted into L. Similarly, we denote by I_i^3 (I_i^4) the number of indices t for which $u^{2t-1} \notin V_i$, $u^{2t} \in V_i$, and u^{2t} (u^{2t-1}) is inserted into L. Clearly, $b_i = (I_i^1 + I_i^3 - I_i^2 - I_i^4)/l$. We will now give an estimate on I_i^1 for some fixed i. Let X_t be an indicator variable for the event that $u^{2t-1} \in V_i$, $u^{2t} \notin V_i$, and u^{2t-1} is inserted into L. Let $p_j = p_{ij}^{>}(f) + \frac{1}{2} p_{ij}^{=}(f)$. Then,

$$\begin{aligned}
\mathrm{P}\left[X_t = 1\right] &= a_i^{2t-1} \sum_{j \neq i} a_j^{2t} p_j \\
&= a_i \sum_{j \neq i} a_j p_j + a_i \sum_{j \neq i} (a_j^{2t} - a_j) p_j + (a_i^{2t-1} - a_i) \sum_{j \neq i} a_j^{2t} p_j \\
&\leq a_i \sum_{j \neq i} a_j p_j + a_i \sum_{j=1}^{m} |a_j^{2t} - a_j| + |a_i^{2t-1} - a_i|
\end{aligned}$$

and using the above bounds on $|a_j - a_j^r|$ we obtain

$$\leq a_i \sum_{j \neq i} a_j p_j + O(a_i \sum_{j=1}^{m} \sqrt{a_j n^{-1} \log n} + \sqrt{a_i n^{-1} \log n}).$$

Since $\sum_{j=1}^{m} a_j = 1$, we have that the maximum value of $\sum_{j=1}^{m} \sqrt{a_j}$ is obtained when $a_1 = a_2 = \cdots = a_m = 1/m$, so $\sum_{j=1}^{m} \sqrt{a_j} \leq \sqrt{m}$. Therefore,

$$\mathrm{P}\left[X_t = 1\right] \leq a_i \sum_{j \neq i} a_j p_j + O(\sqrt{a_i n^{-1} \log n}(1 + \sqrt{a_i m})).$$

Define the quantity in the last equation by θ. As $\mathrm{P}\left[X_t = 1\right] \leq \theta$ for all t, we have that the random variable $I_i^1 = \sum_{t=1}^{l} X_t$ is dominated by a random variable Y, where Y has binomial distribution with l experiments and success probability $\min(1, \theta)$. Since $k \geq \Omega(\log n)$, we have that $\sqrt{a_i n^{-1} \log n} = a_i \sqrt{(a_i n)^{-1} \log n} \leq a_i \sqrt{k^{-1} \log n} \leq O(a_i)$ and $a_i \sqrt{m n^{-1} \log n} \leq a_i \sqrt{(n/k) n^{-1} \log n} \leq O(a_i)$. Therefore, $\theta \leq O(a_i)$, so $\mathrm{E}[Y] \leq l\theta \leq O(a_i l)$. Thus, by Corollary 2.5, we have w.h.p. that

$$\begin{aligned} Y &\leq \mathrm{E}[Y] + O(\sqrt{\mathrm{E}[Y] \log n}) \leq l\theta + O(\sqrt{a_i l \log n}) \\ &\leq l a_i \sum_{j \neq i} a_j p_j + O(\sqrt{a_i l \log n}(1 + \sqrt{a_i m})). \end{aligned}$$

It follows that w.h.p. $I_i^1 \leq l a_i \sum_{j \neq i} a_j p_j + O(\sqrt{a_i l \log n}(1 + \sqrt{a_i m}))$.

Similar arguments give a lower bound on I_i^1, and lower and upper bounds on I_i^2,I_i^3, and I_i^4, namely w.h.p.,

$$\begin{aligned} |I_i^l - l a_i \sum_{j \neq i} a_j (p_{ij}^{>} + \frac{1}{2} p_{ij}^{=})| &\leq O(\sqrt{a_i l \log n}(1 + \sqrt{a_i m})) \qquad \text{for } l = 1, 3 \\ |I_i^l - l a_i \sum_{j \neq i} a_j (p_{ij}^{<} + \frac{1}{2} p_{ij}^{=})| &\leq O(\sqrt{a_i l \log n}(1 + \sqrt{a_i m})) \qquad \text{for } l = 2, 4 \end{aligned}$$

Combining the above inequalities gives the desired bound on b_i. ■

In the next sections, we use procedure Partition in a slightly different scenario than in Theorem 4.1: We have a random set S of vertices, and then the set T is randomly chosen from $V - S$. It is easy to verify that the result of Theorem 4.1 is also valid in that scenario.

We now give two cluster functions, which will be used later in our algorithms. The first function, given in Lemma 4.2, will be used for building initial sets L_0, R_0 with large imbalance. The second function, in Lemma 4.3, will be used to iteratively build a series of sets L_i, R_i with increasing imbalance. Again, in both cases, assume T is a random subset of V.

Lemma 4.2. *Let u be some vertex from $V_1 - T$, and define $f(v) = d_{\{u\}}(v)$ for all $v \in T$. Then $c_1(f) = 2a_1(1 - a_1)\Delta$ and $c_i(f) = -2a_1 a_i \Delta$ for all $i > 1$.*

Proof. Clearly,

$$\begin{aligned} p^{>}_{1j} &= \mathrm{P}\left[f(v) = 1, f(w) = 0 | v \in V_1, w \in V_j\right] \\ &= \mathrm{P}\left[(u, v) \in E, (u, w) \notin E | v \in V_1, w \in V_j\right] = p(1 - r) \end{aligned}$$

and $p^{<}_{1j} = (1 - p)r$, so

$$c_1(f) = 2a_1 \sum_{j \neq i} a_j (p(1 - r) - (1 - p)r) = 2a_1(1 - a_1)\Delta.$$

For $i, j > 1$, $p^{>}_{ij} = p^{<}_{ij} = r(1 - r)$, $p^{>}_{i1} = r(1 - p)$, and $p^{<}_{i1} = (1 - r)p$. Therefore, $c_i(f) = -2a_1 a_i \Delta$. ∎

Lemma 4.3. *Let L' and R' be two disjoint subsets of $V - T$ of size l each, and define $f(v) = d_{L'}(v) - d_{R'}(v)$ for all $v \in T$. Let $b_i = \mathrm{I}(V_i, L', R')$. Suppose that $b_1 \geq b_2 \geq \cdots \geq b_m$, $p, r \in [\frac{1}{4}, \frac{3}{4}]$, $k \geq \Omega(\sqrt{n \log n})$, and $b_1 - b_m \leq \alpha/\Delta\sqrt{l}$ where $\alpha \leq 1$. Furthermore, suppose that the process of building L', R' was independent of the edges between $L' \cup R'$ and T. Let $\beta = \Gamma + \sqrt{(1/m + \Gamma)l^{-1} \log n}$ and $\gamma = \sqrt{\frac{2}{\pi}/(\frac{1}{m}p(1 - p) + (1 - \frac{1}{m})r(1 - r))}$. Then, w.h.p.,*

1. $c_1(f) \geq \gamma\Delta\sqrt{l}a_1(1 - O(\beta\Delta) - \frac{2}{9}\alpha^2)(b_1 - \sum_{j=1}^m a_j b_j) - 3\frac{a_1}{\sqrt{l}}$.
2. $c_i(f) \leq \gamma\Delta\sqrt{l}a_i \left(b_i - \sum_{j=1}^m a_j b_j + (O(\beta\Delta) + \frac{2}{9}\alpha^2)(b_1 - b_m)\right) + 3\frac{a_i}{\sqrt{l}}$ *for all* $i > 1$.
3. $c_i(f) \geq \gamma\Delta\sqrt{l}a_i \left(b_i - \sum_{j=1}^m a_j b_j - (O(\beta\Delta) + \frac{2}{9}\alpha^2)(b_1 - b_m)\right) - 3\frac{a_i}{\sqrt{l}}$ *for all* $i > 1$.

Note that here the notations b_i and l have different meaning that in Theorem 4.1.

Proof. (1) For $j = 1, \ldots, m$, denote $A'_j = |V_j \cap (L' \cup R')|$. We denote $L' = \{u_1, \ldots, u_l\}$ and $R' = \{u_{l+1}, \ldots u_{2l}\}$. We now estimate $p^{>}_{1j} - p^{<}_{1j}$ for some fixed j. Fix some $v \in V_1 \cap T$ and $w \in V_j \cap T$. We have that $f(v) - f(w) = \sum_{i=1}^{4l} X_i$ where $X_1, \ldots, X_{4l}$ are independent random variables with $X_i = d_{\{u_i\}}(v)$ for $i \in [1, l]$, $X_i = -d_{\{u_i\}}(v)$ for $i \in [l + 1, 2l]$, $X_i = -d_{\{u_{i-2l}\}}(w)$ for $i \in [2l + 1, 3l]$, and $X_i = d_{\{u_{i-2l}\}}(w)$ for $i \in [3l + 1, 4l]$. Let $\mathrm{B}(q)$ denote the Bernoulli distribution with parameter q (i.e., a random variable with $\mathrm{B}(q)$ distribution has a value of 1 with probability q, and value of 0 otherwise). Out of the variable $X_1, \ldots, X_l$, there are $\frac{A'_1 + lb_1}{2}$ variables with $\mathrm{B}(p)$ distribution, and the rest of the variables have $\mathrm{B}(r)$ distribution. Out of $X_{l+1}, \ldots, X_{2l}$, $\frac{A'_1 - lb_1}{2}$ variables have $-\mathrm{B}(p)$ distribution, and the rest of the variables have $-\mathrm{B}(r)$ distribution. Similar claims hold for $X_{2l+1}, \ldots, X_{4l}$, and therefore,

$$\begin{aligned}\mathrm{E}[\sum_{i=1}^{4l} X_i] &= p\frac{A_1' + lb_1}{2} + r(l - \frac{A_1' + lb_1}{2}) - p\frac{A_1' - lb_1}{2} - r(l - \frac{A_1' - lb_1}{2}) \\ &\quad - p\frac{A_j' + lb_j}{2} - r(l - \frac{A_j' + lb_j}{2}) + p\frac{A_j' - lb_j}{2} + r(l - \frac{A_j' - lb_j}{2}) \\ &= plb_1 - rlb_1 - plb_j + rlb_j = \Delta l(b_1 - b_j).\end{aligned}$$

Let $B_{1j} = \sum_{i=1}^{4l} \mathrm{Var}(X_i)$. Then $B_{1j} = (A_1' + A_j')p(1-p) + (4l - A_1' - A_j')r(1-r)$, and since $p, r \in [\frac{1}{4}, \frac{3}{4}]$, it follows that $3/16 \leq p(1-p), r(1-r) \leq 1/4$, so $\frac{3}{4}l \leq B_{1j} \leq l$. Denote $h(x) = (1-x)^3 x + x^3(1-x)$. For $L_{1j} = B_{1j}^{-3/2} \sum_{i=1}^{4l} \mathrm{E}[|X_i - \mathrm{E}[X_i]|^3]$ we have that

$$L_{4l} = \frac{1}{B_{1j}^{3/2}}((A_1' + A_j')h(p) + (4l - A_1' - A_j')h(r)) \leq \frac{1}{(\frac{3}{4}l)^{3/2}} \cdot 4l \cdot \frac{1}{8} < \frac{0.77}{\sqrt{l}}.$$

Now, $p_{ij}^{>} = \mathrm{P}\left[\sum_{i=1}^{4l} X_i > 0\right]$ and $p_{ij}^{<} = \mathrm{P}\left[\sum_{i=1}^{4l} X_i < 0\right]$. By Corollary 2.3, $|p_{1j}^{>} - \Phi\left(\frac{\Delta l(b_1 - b_j)}{\sqrt{B_{1j}}}\right)| \leq 0.8 L_{4l} < \frac{3}{4\sqrt{l}}$, and $|p_{1j}^{<} - (1 - \Phi\left(\frac{\Delta l(b_1 - b_j)}{\sqrt{B_{1j}}}\right))| < \frac{3}{4\sqrt{l}}$. We conclude that $p_{1j}^{>} - p_{1j}^{<} \geq 2\Phi\left(\frac{\Delta l(b_1 - b_j)}{\sqrt{B_{1j}}}\right) - 1 - \frac{3}{2\sqrt{l}}$.

For $x \in [0, a]$, where $a \leq 1$, we have $\Phi(x) \geq \frac{1}{2} + \frac{1}{\sqrt{2\pi}}x - \frac{1}{6\sqrt{2\pi}}x^3 \geq \frac{1}{2} + \frac{1}{\sqrt{2\pi}}x(1 - \frac{a^2}{6})$. As $0 \leq \frac{\Delta l(b_1 - b_j)}{\sqrt{B_{1j}}} \leq \frac{\Delta l(b_1 - b_j)}{\sqrt{3l/4}} \leq \frac{2}{\sqrt{3}}\alpha$, then

$$p_{1j}^{>} - p_{1j}^{<} \geq \sqrt{\frac{2}{\pi}} \frac{\Delta l(b_1 - b_j)}{\sqrt{B_{1j}}}(1 - \frac{2}{9}\alpha^2) - \frac{3}{2\sqrt{l}}.$$

Let $B = 4l\frac{1}{m}p(1-p) + 4l(1 - \frac{1}{m})r(1-r)$. By Corollary 2.6, w.h.p. $|A_i' - 2la_i| \leq O(\sqrt{a_i l \log n}) \leq O(\sqrt{(1/m + \Gamma)l \log n}) \leq O(\beta l)$ for all i, so $|A_i' - \frac{2l}{m}| \leq |A_i' - 2la_i| + 2l|a_i - \frac{1}{m}| \leq O(\beta l)$. Therefore,

$$|B_{1j} - B| = |(A_1' + A_j' - \frac{4l}{m})(p(1-p) - r(1-r))| \leq O(\beta l \cdot \Delta |1 - p - r|) \leq O(\beta l \Delta).$$

Since $B, B_{1j} \geq \frac{3}{4}l$, we conclude that $|B_{1j} - B| \leq O(\beta \Delta B_{1j})$ and $|B_{1j} - B| \leq O(\beta \Delta B)$. Thus, for $B \geq B_{1j}$,

$$\left|\frac{1}{\sqrt{B_{1j}}} - \frac{1}{\sqrt{B}}\right| = \frac{1}{\sqrt{B}}(\sqrt{\frac{B}{B_{1j}}} - 1) \leq \frac{1}{\sqrt{B}}(\sqrt{1 + O(\beta\Delta)} - 1) \leq O(\frac{\beta\Delta}{\sqrt{B}})$$

and for $B < B_{1j}$,

$$\left| \frac{1}{\sqrt{B_{1j}}} - \frac{1}{\sqrt{B}} \right| = \frac{1}{\sqrt{B_{1j}}} (\sqrt{\frac{B_{1j}}{B}} - 1) \leq O(\frac{\beta\Delta}{\sqrt{B_{1j}}}) = O(\frac{\beta\Delta}{\sqrt{B}}).$$

Therefore,

$$\begin{aligned}
p^{>}_{1j} - p^{<}_{1j} &\geq \sqrt{\frac{2}{\pi}} \frac{\Delta l(b_1 - b_j)}{\sqrt{B}} (1 - O(\beta\Delta))(1 - \frac{2}{9}\alpha^2) - \frac{3}{2\sqrt{l}} \\
&\geq \sqrt{\frac{2}{\pi}} \frac{\Delta l(b_1 - b_j)}{\sqrt{B}} (1 - O(\beta\Delta) - \frac{2}{9}\alpha^2) - \frac{3}{2\sqrt{l}} \\
&= \frac{\gamma}{2} \Delta \sqrt{l} (b_1 - b_j)(1 - O(\beta\Delta) - \frac{2}{9}\alpha^2) - \frac{3}{2\sqrt{l}},
\end{aligned}$$

and

$$\begin{aligned}
c_1(f) &\geq 2a_1 \sum_{j=1}^{m} a_j \frac{\gamma}{2} \Delta \sqrt{l} (b_1 - b_j)(1 - O(\beta\Delta) - \frac{2}{9}\alpha^2) - a_1 \sum_{j \neq 1} a_j \frac{3}{\sqrt{l}} \\
&\geq \gamma \Delta \sqrt{l} a_1 (1 - O(\beta\Delta) - \frac{2}{9}\alpha^2)(b_1 - \sum_{j=1}^{m} a_j b_j) - 3\frac{a_1}{\sqrt{l}}.
\end{aligned}$$

(2) Like in case (1), we have that for every $j \neq i$, $p^{>}_{ij} - p^{<}_{ij} \leq 2\Phi(\frac{\Delta l(b_i - b_j)}{\sqrt{B_{ij}}}) - 1 + \frac{3}{2\sqrt{l}}$ where $B_{ij} = (A'_i + A'_j)p(1-p) + (4l - A'_i - A'_j)r(1-r)$. For $x \in [0,1]$, $\Phi(x) \leq \frac{1}{2} + \frac{1}{\sqrt{2\pi}} x$, so for $j > i$,

$$\begin{aligned}
p^{>}_{ij} - p^{<}_{ij} &\leq \sqrt{\frac{2}{\pi}} \frac{\Delta l(b_i - b_j)}{\sqrt{B_{ij}}} + \frac{3}{2\sqrt{l}} \\
&\leq \sqrt{\frac{2}{\pi}} \frac{\Delta l(b_i - b_j)}{\sqrt{B}} (1 + O(\beta\Delta)) + \frac{3}{2\sqrt{l}} \\
&= \frac{\gamma}{2} \Delta \sqrt{l} (b_i - b_j)(1 + O(\beta\Delta)) + \frac{3}{2\sqrt{l}}.
\end{aligned}$$

Furthermore, for $x \in [-a, 0]$ (where $a \leq 1$), $\Phi(x) \leq \frac{1}{2} + \frac{1}{\sqrt{2\pi}} x(1 - \frac{a^2}{6})$, so for $j < i$,

$$\begin{aligned}
p^{>}_{ij} - p^{<}_{ij} &\leq \sqrt{\frac{2}{\pi}} \frac{\Delta l(b_i - b_j)}{\sqrt{B_{ij}}} (1 - \frac{2}{9}\alpha^2) + \frac{3}{2\sqrt{l}} \\
&\leq \sqrt{\frac{2}{\pi}} \frac{\Delta l(b_i - b_j)}{\sqrt{B}} (1 - O(\beta\Delta))(1 - \frac{2}{9}\alpha^2) + \frac{3}{2\sqrt{l}} \\
&\leq \frac{\gamma}{2} \Delta \sqrt{l} (b_i - b_j)(1 - O(\beta\Delta) - \frac{2}{9}\alpha^2) + \frac{3}{2\sqrt{l}}.
\end{aligned}$$

Therefore,

$$
\begin{aligned}
c_i(f) \le 2a_i \Bigg(& \sum_{j=1}^{i-1} a_j \frac{\gamma}{2}\Delta\sqrt{l}(b_i - b_j)(1 - O(\beta\Delta) - \frac{2}{9}\alpha^2) \\
& + \sum_{j=i+1}^{m} a_j \frac{\gamma}{2}\Delta\sqrt{l}(b_i - b_j)(1 + O(\beta\Delta)) + \sum_{j\ne i} a_j \frac{3}{2\sqrt{l}} \Bigg) \\
\le \gamma\Delta\sqrt{l}a_i & \left(b_i - \sum_{j=1}^{m} a_j b_j + (O(\beta\Delta) + \frac{2}{9}\alpha^2) \sum_{j=1}^{m} a_j |b_i - b_j| \right) + 3\frac{a_i}{\sqrt{l}} \\
\le \gamma\Delta\sqrt{l}a_i & \left(b_i - \sum_{j=1}^{m} a_j b_j + (O(\beta\Delta) + \frac{2}{9}\alpha^2)(b_1 - b_m) \right) + 3\frac{a_i}{\sqrt{l}}.
\end{aligned}
$$

(3) We have, $p^{>}_{ij} - p^{<}_{ij} \ge \frac{\gamma}{2}\Delta\sqrt{l}(b_i - b_j)(1 - O(\beta\Delta) - \frac{2}{9}\alpha^2) - \frac{2}{\sqrt{l}}$ for $j > i$, and $p^{>}_{ij} - p^{<}_{ij} \ge \frac{\gamma}{2}\Delta\sqrt{l}(b_i - b_j)(1 + O(\beta\Delta)) - \frac{2}{\sqrt{l}}$ for $j < i$. Therefore,

$$
\begin{aligned}
c_i(f) \ge 2a_i \Bigg(& \sum_{j=1}^{i-1} a_j \frac{\gamma}{2}\Delta\sqrt{l}(b_i - b_j)(1 + O(\beta\Delta)) \\
& + \sum_{j=i+1}^{m} a_j \frac{\gamma}{2}\Delta\sqrt{l}(b_i - b_j)(1 - O(\beta\Delta) - \frac{2}{9}\alpha^2) - \sum_{j\ne i} a_j \frac{3}{2\sqrt{l}} \Bigg) \\
\ge \gamma\Delta\sqrt{l}a_i & \left(b_i - \sum_{j=1}^{m} a_j b_j - (O(\beta\Delta) + \frac{2}{9}\alpha^2) \sum_{j=1}^{m} a_j |b_i - b_j| \right) - 3\frac{a_i}{\sqrt{l}} \\
\ge \gamma\Delta\sqrt{l}a_i & \left(b_i - \sum_{j=1}^{m} a_j b_j - (O(\beta\Delta) + \frac{2}{9}\alpha^2)(b_1 - b_m) \right) - 3\frac{a_i}{\sqrt{l}}. \qquad \blacksquare
\end{aligned}
$$

5 Almost equal sized clusters

In this section we present an algorithm for finding a subcluster in the case where the sizes of all clusters are almost equal: The algorithm requires that $\Gamma \le O(1/m \log n)$. We use algorithm Solve_2 with a procedure Find which is described below.

Procedure Find begins by building two initial sets L_0 and R_0 using procedure Partition and the cluster function of Lemma 4.2. Then a series of sets L_t, R_t is generated, where L_t and R_t are generated from L_{t-1} and R_{t-1} using procedure Partition and the cluster function of Lemma 4.3. The main idea is that the L_t, R_t-imbalance increases at each iteration. In order to stop the process, procedure Find also builds sets $\hat{L}_t$ and $\hat{R}_t$ which are built by the same process in which L_t and R_t

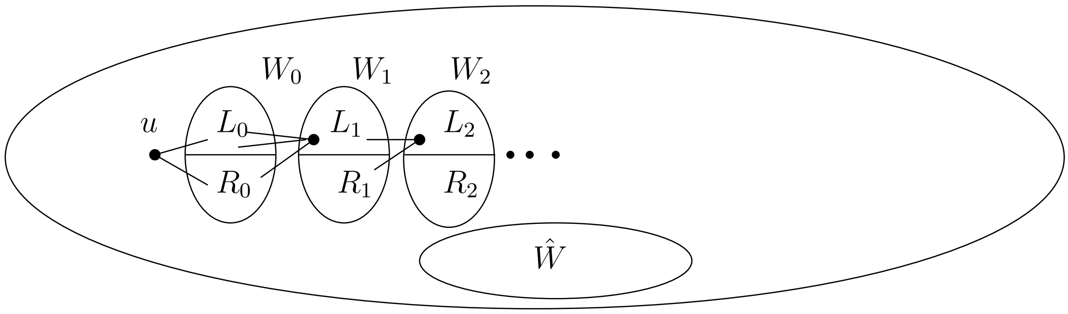

Figure 2: The sets created by procedure Find.

are built. At each iteration, we sort the vertices of some random set of vertices S according to their f-values where $f(v) = d_{\hat{L}_t}(v) - d_{\hat{R}_t}(v)$. If there is a large gap in the sorted f-values, then the procedure stops, and outputs the vertices whose f-values are above the gap.

Formally, procedure Find is as follows:

Find(G, V')

1. Let $\hat{l} = \Theta(m^2\Delta^{-2}\log^2 n + \Delta^2 m^{-1}\log^5 n)$, $l = \Theta(\hat{l}/\log n)$, $l' = \Theta(\hat{l}/\log^2 n)$, $s = \Theta(m\Delta^{-2}\log n)$, and $D = \Theta(\sqrt{\hat{l}\log n})$. Let $l_t = l$ for $0 \le t \le 2$ and $l_t = l'$ for $t \ge 3$.
2. Randomly select disjoint sets of vertices S, $\hat{W}$, and W_0 from V' of sizes s, $2\hat{l}$, and $2l_0$, respectively.
3. Randomly select some unchosen vertex u from V'.
4. $\hat{L}_0, \hat{R}_0 \leftarrow$ Partition$(G, \hat{W}, d_{\{u\}})$ and $L_0, R_0 \leftarrow$ Partition$(G, W_0, d_{\{u\}})$.
5. For $t = 0, 1, \ldots, \log n$ do:
   (a) Let $v_1, \ldots, v_s$ be an ordering of S such that $f(v_1) \ge f(v_2) \ge \cdots \ge f(v_s)$, where $f(v) = d_{\hat{L}_t}(v) - d_{\hat{R}_t}(v)$. If $\max_j\{f(v_j) - f(v_{j+1})\} \ge D$, then let j be the first index for which $f(v_j) - f(v_{j+1}) \ge \frac{1}{3}D$, and return $\{v_1, \ldots, v_j\}$. Otherwise, continue.
   (b) Randomly select a set W_{t+1} from V' of $2l_{t+1}$ unchosen vertices.
   (c) $\hat{L}_{t+1}, \hat{R}_{t+1} \leftarrow$ Partition$(G, \hat{W}, d_{L_t} - d_{R_t})$.
   (d) $L_{t+1}, R_{t+1} \leftarrow$ Partition$(G, W_{t+1}, d_{L_t} - d_{R_t})$.
6. Output 'Failure'.

See also Figure 2. The total number of vertices selected by procedure Find is at most $s + 1 + 2\hat{l} + 2\sum_{t=0}^{\log n+1} l_t$, and this number can be more than $n/3$ for some choice of parameters, making the procedure inapplicable. However, we assume that the bound $k \ge \Omega(\Delta^{-1}\sqrt{n}\log n)$ is satisfied, and in that case, $\Delta^{-1} \le$

$O(k/\sqrt{n}\log n) \le O(\sqrt{n}/m\log n)$. Therefore, $s \le \hat{o}(n/m\log n)$ and $\hat{l} \le \hat{o}(n)$. It follows that the total number of vertices selected by Find is at most $n/3$. The requirement $k \ge \Omega(\Delta^{-1}\sqrt{n}\log n)$ also implies that $n \ge c\sqrt{n}$ for some constant c, so $n \ge c^2$. Assuming that the constant c is large, we obtain that n is large.

Note that in the first three iterations of Step 5, the sets L_t, R_t are bigger than in the rest of the steps. The reason is that in these steps, the imbalance is small, so in order to get a small *relative* difference between the imbalances and the expected imbalances, we need to use bigger sets (see Theorem 4.1). We also note that the $\Theta(\log n)$ or $\Theta(\log^2 n)$ size ratio between $\hat{L}_t, \hat{R}_t$ and L_t, R_t has an important role: It allows us to ensure that the requirement $|b'_i - b'_j| \le \alpha/\Delta\sqrt{l'}$ of Lemma 4.3 is satisfied for a small α.

Denote $b_i^t = \mathrm{I}(V_i, L_t, R_t)$, $\hat{b}_i^t = \mathrm{I}(V_i, \hat{L}_t, \hat{R}_t)$, $r_i^t = b_i^t/b_1^t$, and $\hat{r}_i^t = \hat{b}_i^t/\hat{b}_1^t$. In the following, we assume w.l.o.g. that the vertex u chosen in Step 3 is from V_1. Let t^* be the value of t when the procedure Find stops. For simplicity, we assume that $t^* \ge 4$.

Lemma 5.1. *Suppose that $\Gamma \le O(1/m\log n)$, $k \ge \Omega(\Delta^{-1}\sqrt{n}\log n)$, and $p, r \in [\frac{1}{4}, \frac{3}{4}]$. Then, w.h.p.,*

1. *For all i, $|b_i^t - \hat{b}_i^t| \le \hat{o}(\frac{\Delta}{m})$ for $0 \le t \le 2$, and $|b_i^t - \hat{b}_i^t| \le \hat{o}(\frac{\Delta}{m}\sqrt{\log n})$ for $3 \le t \le t^*$.*
2. *$b_1^0, \hat{b}_1^0 \ge (1 - \hat{o}(1))\frac{\Delta}{m}$.*
3. *If $m \ge 3$, then for all $i > 1$, $-\frac{1}{2} - \hat{o}(1) \le r_i^0, \hat{r}_i^0 \le \hat{o}(1)$.*
4. *For $1 \le t \le 3$, $b_1^t, \hat{b}_1^t \ge \log^{t/2} n \cdot \frac{\Delta}{m}$.*
5. *For $4 \le t \le t^*$, $b_1^t, \hat{b}_1^t \ge 2^{t-3}\log^{3/2} n \cdot \frac{\Delta}{m}$.*
6. *If $m \ge 3$, then for all $1 \le t \le 3$ and all $i > 1$,*
$$(1+\delta)^t(\min(0, r_i^0) - \delta t) \le r_i^t \le (1+\delta)^t(\max(0, r_i^0) + \delta t)$$
and
$$(1+\delta)^t(\min(0, \hat{r}_i^0) - \delta t) \le \hat{r}_i^t \le (1+\delta)^t(\max(0, \hat{r}_i^0) + \delta t)$$
where $\delta \le \hat{o}(1)$.
7. *If $m \ge 3$, then for all $4 \le t \le t^*$ and all $i > 1$,*
$$(1+\delta')^{t-3}(\min(0, r_i^3) - \delta'(t-3)) \le r_i^t \le (1+\delta')^{t-3}(\max(0, r_i^3) + \delta'(t-3))$$
and
$$(1+\delta')^{t-3}(\min(0, \hat{r}_i^3) - \delta'(t-3)) \le \hat{r}_i^t \le (1+\delta')^{t-3}(\max(0, \hat{r}_i^3) + \delta'(t-3))$$
where $\delta' \le \hat{o}(1/\log n)$.

Note that as $t^* \leq \log n$, we have from (6) and (7) that $-\frac{1}{2} - \hat{o}(1) \leq r_i^t \leq \hat{o}(1)$ for all t when $m \geq 3$. Furthermore, in the case $m = 2$, we have by definition that $b_2^t = -b_1^t$ and $\hat{b}_2^t = -\hat{b}_1^t$ for all t, so $r_2^t, \hat{r}_2^t = -1$ for all t.

Proof. In the proof of items 2–7, we only prove the bounds on b_i^t and r_i^t, as the bounds on $\hat{b}_i^t$ and $\hat{r}_i^t$ are similar. The lemma is proved using induction on t.

(1) Denote $c_i^0 = c_i(d_{\{u\}})$ and $c_i^t = c_i(d_{L_t} - d_{R_t})$ for $t \geq 1$. By Theorem 4.1 we have w.h.p. that $|b_i^t - c_i^t| \leq d_i$ and $|\hat{b}_i^t - c_i^t| \leq d_i$ for all i and $t \leq 2$, where

$$d_i \leq O(\sqrt{\frac{a_i}{l}\log n}(1 + \sqrt{a_i m})) = O(\sqrt{\frac{1}{ml}\log n}) \leq \hat{o}(\frac{\Delta}{m}).$$

(note that the difference between $\hat{b}_i^t$ and c_i^t is even smaller). Furthermore, for $3 \leq t \leq t^*$ we have w.h.p. that $|b_i^t - c_i^t| \leq d_i'$ and $|\hat{b}_i^t - c_i^t| \leq d_i'$ for all i, where

$$d_i' \leq O(\sqrt{\frac{a_i}{l'}\log n}(1 + \sqrt{a_i m})) = O(\sqrt{\frac{1}{ml'}\log n}) \leq \hat{o}(\frac{\Delta}{m}\sqrt{\log n}).$$

Throughout this proof we assume that these inequalities hold. In particular, for every i, $|b_i^t - \hat{b}_i^t| \leq 2d_1$ for $t \leq 2$ and $|b_i^t - \hat{b}_i^t| \leq 2d_1'$ for $3 \leq t \leq t^*$.

(2) From Lemma 4.2, we have that

$$c_1^0 = 2a_1(1-a_1)\Delta \geq 2(1-o(1))\frac{1}{m}(1-(1+o(1))\frac{1}{m})\Delta \geq (1-o(1))\frac{\Delta}{m}.$$

Therefore, $b_1^0 \geq c_1^0 - d_1 \geq (1-\hat{o}(1))\frac{\Delta}{m}$. Note that for $m \geq 3$ we have that $b_1^0 \geq (1-\hat{o}(1))\frac{4}{3}\frac{\Delta}{m}$.

(3) By Lemma 4.2, $c_i^0 = -2a_1 a_i \Delta$ for every $i > 1$. Therefore, $b_i^0 \leq c_i^0 + d_i \leq d_i \leq \hat{o}(\frac{\Delta}{m})$. Thus, $r_i^0 \leq \hat{o}(1)$. Furthermore, $c_i^0 \geq -2((1+o(1))\frac{1}{m})^2\Delta \geq -(1+o(1))\frac{2}{3}\frac{\Delta}{m}$, so $b_i^0 \geq c_i^0 - d_i \geq -(1+\hat{o}(1))\frac{2}{3}\frac{\Delta}{m}$. Thus, $r_i^0 \geq -(1+\hat{o}(1))\frac{2}{3}\frac{\Delta}{m}/((1-\hat{o}(1))\frac{4}{3}\frac{\Delta}{m}) = -\frac{1}{2} - \hat{o}(1)$.

(4) As $b_1^0 \geq \frac{1}{2}\frac{\Delta}{m}$, it suffices to prove that $b_1^{t+1} \geq 2\sqrt{\log n} \cdot b_1^t$ for all $0 \leq t \leq 2$. Fix some $0 \leq t \leq 2$. For the rest of the proof, we assume w.l.o.g. that $b_1^t \geq b_2^t \geq \cdots \geq b_m^t$. By Corollary 2.6, w.h.p., $||V_i \cap S| - a_i s| \leq O(\sqrt{a_i s \log n}) = O(\sqrt{(1/m) \cdot s \log n}) \leq 1/2 \cdot s/m$ for all i, hence $|V_i \cap S| \geq a_i s - 1/2 \cdot s/m = \Omega(s/m) = \Omega(\log n/\Delta^2)$. In particular, $V_1 \cap S \neq \phi$ and $S - V_1 \neq \phi$.

By Corollary 2.5, w.h.p., $|f(v) - \mathrm{E}[f(v)]| < \frac{1}{2}D$ for all $v \in S$. From the proof of Lemma 4.3, we have that for a vertex $v \in V_i \cap S$, $\mathrm{E}[f(v)] = \Delta\hat{l}\hat{b}_i^t$. Since the algorithm did not stop at iteration t, we have that $\Delta\hat{l}(\hat{b}_1^t - \hat{b}_2^t) \leq 2D$, because otherwise, the difference between the minimum value of $f(v)$ for $v \in V_1 \cap S$ and the maximum value of $f(v')$ for $v' \in S - V_1$ would be at least D. From (1) and (6) we have that for all i, j, $|b_i^t - b_j^t| \leq |\hat{b}_i^t - \hat{b}_j^t| + \hat{o}(\frac{\Delta}{m}) \leq 2(\hat{b}_1^t - \hat{b}_2^t) + \hat{o}(\frac{\Delta}{m})$. By induction, $\hat{b}_1^t - \hat{b}_2^t \geq (1-\hat{o}(1))\hat{b}_1^t \geq \Omega(\frac{\Delta}{m})$. Therefore, for all i, j, $|b_i^t - b_j^t| \leq$

$(2+\hat{o}(1))(\hat{b}_1^t-\hat{b}_2^t) \leq O(D/\Delta\hat{l}) = O(\sqrt{\log n}/\Delta\sqrt{\hat{l}})$. As $\hat{l} = \Theta(l \log n)$, we conclude that for all i, j, $|b_i^t - b_j^t| \leq \alpha/\Delta\sqrt{l}$ where $\alpha \leq \hat{o}(1)$.

Denote $F_0 = \gamma\Delta\sqrt{l}$ (where γ is defined in Lemma 4.3), and $F = F_0/m$. By Lemma 4.3,

$$\begin{aligned} b_1^{t+1} &\geq F_0 a_1(1 - O(\Gamma\Delta + \sqrt{\frac{1}{ml}\log n}\Delta) - \frac{2}{9}\alpha^2)(b_1^t - \sum_{j=1}^m a_j b_j^t) - 3\frac{a_1}{\sqrt{l}} - d_1 \\ &\geq F_0(1-o(1))\frac{1}{m}(1-\hat{o}(1))(b_1^t - \sum_{j=1}^m a_j b_j^t) - 3\frac{a_1}{\sqrt{l}} - d_1 \\ &= (1-\hat{o}(1))F(b_1^t - \sum_{j=1}^m a_j b_j^t) - 3\frac{a_1}{\sqrt{l}} - d_1. \end{aligned}$$

From (6) we have that $|b_j^t| \leq b_1^t$ for all j, and therefore, using the fact that $\sum_{j=1}^m b_j^t = 0$, we have that

$$\sum_{j=1}^m a_j b_j^t = \sum_{j=1}^m (a_j - \frac{1}{m})b_j^t \leq \sum_{j=1}^m |a_j - \frac{1}{m}|b_1^t \leq m\Gamma b_1^t \leq \hat{o}(\frac{1}{\log n}b_1^t).$$

Furthermore, from the fact that $F = \Omega(\Delta\sqrt{l}/m) \geq \Omega(\sqrt{\log n})$ and $b_1^t \geq \Omega(\Delta/m)$ we obtain

$$3\frac{a_1}{\sqrt{l}} \leq O(\frac{1}{\sqrt{l}}) \leq O(\frac{1}{\sqrt{\log n}}\frac{\Delta}{m}) \leq o(Fb_1^t),$$

and

$$d_1 \leq \hat{o}(\frac{\Delta}{m}) \leq o(Fb_1^t).$$

Therefore, $b_1^{t+1} \geq (1-\hat{o}(1))Fb_1^t$. In particular, $b_1^{t+1} \geq 2\sqrt{\log n} \cdot b_1^t$.

(5) It suffices to prove that $b_1^{t+1} \geq 2b_1^t$ for all $3 \leq t \leq t^* - 1$. Similarly to the proof of (4), we have that for all i, j, $|b_i^t - b_j^t| \leq O(\sqrt{\log n}/\Delta\sqrt{\hat{l}})$, and since $\hat{l} = \Theta(l' \log^2 n)$, we conclude that $|b_i^t - b_j^t| \leq \alpha_2/\Delta\sqrt{l'}$ where $\alpha_2 \leq \hat{o}(1/\sqrt{\log n})$.

Denote $F_0' = \gamma\Delta\sqrt{l'}$, and $F' = F_0'/m$. By Lemma 4.3,

$$\begin{aligned} b_1^{t+1} &\geq F_0' a_1(1 - O(\Gamma\Delta + \sqrt{\frac{1}{ml'}\log n}\Delta) - \frac{2}{9}\alpha_2^2)(b_1^t - \sum_{j=1}^m a_j b_j^t) - 3\frac{a_1}{\sqrt{l'}} - d_1' \\ &\geq (1 - \hat{o}(\frac{1}{\log n}))F' b_1^t - 3\frac{a_1}{\sqrt{l'}} - d_1'. \end{aligned}$$

Furthermore,

$$3\frac{a_1}{\sqrt{l'}} \leq O(\frac{1}{\sqrt{l'}}) \leq O(\frac{\Delta}{m}) \leq o(\frac{1}{\log n}F' b_1^t),$$

where the last equality follows from the fact that $F' = \Omega(\Delta\sqrt{l'}/m) \geq \Omega(1)$ and $b_1^t \geq \log^{3/2} n \cdot \Delta/m$. We also have

$$d_1' \leq \hat{o}(\frac{\Delta}{m}\sqrt{\log n}) \leq \hat{o}(\frac{1}{\log n}F'b_1^t).$$

Therefore, $b_1^{t+1} \geq (1 - \hat{o}(1/\log n))F'b_1^t$. In particular, $b_1^{t+1} \geq 2b_1^t$.

(6) We begin by giving an upper bound on b_i^{t+1}. By Lemma 4.2 and the fact that $\sum_{j=1}^m a_j b_j^t \geq -m\Gamma b_1^t \geq -o(b_1^t)$,

$$\begin{aligned}
b_i^{t+1} &\leq F_0 a_i(b_i^t - \sum_{j=1}^m a_j b_j^t + (O(\Gamma\Delta + \sqrt{\frac{1}{ml}\log n\Delta}) + \frac{2}{9}\alpha^2)(b_1^t - b_m^t)) + 3\frac{a_i}{\sqrt{l}} + d_i \\
&\leq F_0 a_i(b_i^t + o(b_1^t) + \hat{o}(1)(b_1^t - b_m^t)) + o(Fb_1^t) \\
&\leq Fb_i^t + o(1)F|b_i^t| + \hat{o}(1)Fb_1^t + \hat{o}(1)F(b_1^t - b_m^t).
\end{aligned}$$

Since $|b_i^t| \leq b_1^t$ and $b_1^t - b_m^t \leq 2b_1^t$, it follows that

$$b_i^{t+1} \leq Fb_i^t + \hat{o}(1)Fb_1^t = (r_i^t + \hat{o}(1))Fb_1^t \leq (\max(0, r_i^t) + \hat{o}(1))Fb_1^t.$$

Thus, we have that $b_1^{t+1} \geq (1 - \delta_0)Fb_1^t$ and $b_i^{t+1} \leq (\max(0, r_i^t) + \delta_0)Fb_i^t$ for $\delta_0 \leq \hat{o}(1)$. Hence, for $\delta = 2\delta_0$ we have

$$\begin{aligned}
r_i^{t+1} &\leq \frac{\max(0, r_i^t) + \delta_0}{1 - \delta_0} \leq (1 + \delta)\max(0, r_i^t) + \delta \\
&\leq (1 + \delta)(1 + \delta)^t(\max(0, r_i^0) + \delta t) + \delta \\
&\leq (1 + \delta)^{t+1}(\max(0, r_i^0) + (t + 1)\delta).
\end{aligned}$$

The lower bound on r_i^{t+1} is shown using similar arguments: We have that

$$\begin{aligned}
b_i^{t+1} &\geq F_0 a_i(b_i^t - \sum_{j=1}^m a_j b_j^t - (O(\Gamma\Delta + \sqrt{\frac{1}{ml}\log n\Delta}) + \frac{2}{9}\alpha^2)(b_1^t - b_m^t)) - 3\frac{a_i}{\sqrt{l}} - d_i \\
&\geq Fb_i^t - \hat{o}(1)Fb_1^t \geq (\min(0, r_i^t) - \hat{o}(1))Fb_1^t.
\end{aligned}$$

Therefore,

$$r_i^{t+1} \geq \frac{\min(0, r_i^t) - \delta_0}{1 - \delta_0} \geq (1 + \delta)\min(0, r_i^t) - \delta \geq (1 + \delta)^{t+1}\min(0, r_i^0) - (t + 1)\delta.$$

(7) By Lemma 4.2,

$$\begin{aligned}
b_i^{t+1} &\leq F_0' a_i(b_i^t - \sum_{j=1}^m a_j b_j^t + (O(\Gamma\Delta + \sqrt{\frac{1}{ml'}\log n\Delta}) + \frac{2}{9}\alpha_2^2)(b_1^t - b_m^t)) + 3\frac{a_i}{\sqrt{l'}} + d_i' \\
&\leq F'b_i^t + \hat{o}(\frac{1}{\log n})F'b_1^t \leq (\max(0, r_i^t) + \hat{o}(\frac{1}{\log n}))F'b_1^t.
\end{aligned}$$

Thus, we have that $b_1^{t+1} \geq (1 - \delta_0')F'b_1^t$ and $b_i^{t+1} \leq (\max(0, r_i^t) + \delta_0')F'b_i^t$ for $\delta_0' \leq \hat{o}(1/\log n)$. Therefore, $r_i^{t+1} \leq (1 + \delta')^{t-3}(\max(0, r_i^3) + \delta'(t - 3))$ for $\delta' = 2\delta_0'$. The lower bound is proved in a similar manner. ∎

Lemma 5.2. *Suppose that $\Gamma \le O(1/m \log n)$, $k \ge \Omega(\Delta^{-1}\sqrt{n}\log n)$, and $p, r \in [\frac{1}{4}, \frac{3}{4}]$. Then, w.h.p., procedure* Find *returns a subcluster of size $\Omega(m \log n/\Delta^2)$. The running time of procedure* Find *is $O(m^3\Delta^{-4}\log^3 n \cdot (m + \log n) + \log^9 n)$.*

Proof. By Lemma 5.1, for all $4 \le t \le t^*$, $b_1^t \ge 2^{t-3}\log^{3/2} n \cdot \frac{\Delta}{m} > 2^t \cdot \frac{\Delta}{m}$. As $m \le n/k \le O(\Delta\sqrt{n}/\log n)$, we have that $b_1^t \ge \Omega(2^t \log n/\sqrt{n}) > 2^t/n$. Since $b_1^t \le 1$ by definition, we conclude that $t^* < \log n$, namely procedure Find stops at Step 5a. We now look at the values of f at the last iteration. W.l.o.g. assume that $\hat{b}_1^{t^*} \ge \hat{b}_2^{t^*} \ge \cdots \ge \hat{b}_m^{t^*}$. From the proof of Lemma 5.1 (4), $V_i \cap S \ne \phi$ for all i. By Corollary 2.5, w.h.p., $|f(v) - \mathrm{E}[f(v)]| < \frac{1}{6}D$ for all $v \in S$, where $\mathrm{E}[f(v)] = \Delta\hat{l}\hat{b}_i^{t^*}$ for $v \in V_i$. As $\max_j\{f(v_j) - f(v_{j+1})\} \ge D$, we have that there is an index i such that $\Delta\hat{l}\hat{b}_i^{t^*} - \Delta\hat{l}\hat{b}_{i+1}^{t^*} \ge \frac{2}{3}D$. By Lemma 5.1, $\Delta\hat{l}(\hat{b}_1^{t^*} - \hat{b}_2^{t^*}) \ge \Delta\hat{l}(\hat{b}_i^{t^*} - \hat{b}_{i+1}^{t^*}) \ge \frac{2}{3}D$. Therefore, $\min_{v \in V_1 \cap S} f(v) - \max_{v \in S - V_1} f(v) > \frac{1}{3}D$. Furthermore, for $v, v' \in V_1 \cap S$, $|f(v) - f(v')| < \frac{1}{3}D$. It follows that $\{v_1, \ldots, v_j\} = V_1 \cap S$, and in particular, $\{v_1, \ldots, v_j\}$ is a subcluster. Finally, it was shown in the proof of Lemma 5.1 (4) that $|V_1 \cap S| = \Omega(\log n/\Delta^2)$.

We now compute the time complexity of Find: In a single iteration, the time to compute $f(v)$ for all $v \in S$ is $O(s\hat{l})$, and the time taken by the call to Partition is $O(l_t\hat{l})$. Therefore, the time complexity of Find is

$$\begin{aligned} O(s\hat{l}\log n + l\hat{l} + l'\hat{l}\log n) &= O(\frac{m^3}{\Delta^4}\log^4 n + \log^7 n + \frac{m^4}{\Delta^{-4}}\log^3 n + \frac{\Delta^4}{m^2}\log^9 n) \\ &= O(\frac{m^3}{\Delta^4}\log^3 n \cdot (m + \log n) + \log^9 n). \end{aligned}$$ ∎

Combining Lemma 5.2 and Lemma 3.2 gives the following theorem:

Theorem 5.3. *The above algorithm solves the clustering problem w.h.p. when $\Gamma \le O(1/m \log n)$ and $k \ge \Omega(\Delta^{-1}\sqrt{n}\log n)$. The running time of the algorithm is $O(m^4\Delta^{-4}\log^3 n \cdot (m + \log n) + m\Delta^{-2}n\log n)$.*

As noted above, the requirement $k \ge \Omega(\Delta^{-1}\sqrt{n}\log n)$ implies that $\Delta^{-1} \le O(\sqrt{n}/m \log n)$. Therefore, $m^4\Delta^{-4}\log^3 n \le O(n^2/\log n)$ and $m\Delta^{-2}n\log n \le O(n^2/m\log n)$. It follows that the time complexity of the algorithm is $O((m/\log n + 1)n^2)$.

Note that algorithm requires the knowledge of m and Δ. However, if we use take $\hat{l} = \Theta(M^2\log^2 n + \log^5 n)$ and $s = \Theta(M^2 \log n)$ where $M \ge m/\Delta$, then procedure Find remains correct, and its time complexity becomes $O(M^4\log^4 n + \log^9 n)$. Since $m/\Delta \le O(\sqrt{n}/\log n)$, one can use $M = \Theta(\sqrt{n}/\log n)$, and the time complexity of procedure Find will be $O(n^2)$. The overall time complexity of the algorithm in this case is $O(mn^2)$.

6 Unequal sized clusters

In this section we give an algorithm for finding a subcluster in the general case. We use algorithm Solve$_3$ and procedure Find$_2$ which is similar to procedure Find that was given in Section 5.

We first describe the main differences between procedures Find$_2$ and Find. First, unlike procedure Find, in which the vertex u was chosen randomly, in procedure Find$_2$ the vertex u must be chosen in a way that ensures that the size of the cluster that contains u is almost equal to the size of the largest cluster. This is done in the following way: Procedure Find$_2$ randomly chooses two sets of vertices S_1 and S_2, and then selects u to be the vertex from S_1 with highest d_{S_2}-value. A second difference comes from the fact that our analysis in the general case is less tight than the analysis in the case of almost equal sized clusters. As a result, procedure Find$_2$ can perform only a constant number of iterations. This leads to a different choice of parameters (namely, the sizes of the sets built by the algorithm).

Procedure Find$_2$ is as follows:

Find$_2(G, V')$

1. Let $\epsilon = \Theta(1)$, $c = \Theta(1)$, $\hat{l} = \Theta(m\Delta^{-2}\log^2 n + m^2\Delta^{-2(1+\epsilon)}\log n)$, $l = \Theta(\hat{l}/\log n)$, $s = \Theta(m\Delta^{-2}\log n)$, $s_1 = \Theta(m \log n)$, $s_2 = \Theta(n)$, and $D = \Theta(\sqrt{\hat{l}\log n})$.
2. Randomly select disjoint sets of vertices S_1, S_2, S, $\hat{W}$, and W_0 from V' of sizes s, s_1, s_2, $2\hat{l}$, and $2l$, respectively.
3. Let u be a vertex in S_1 such that $d_{S_2}(u)$ is maximum.
4. $\hat{L}_0, \hat{R}_0 \leftarrow \text{Partition}(G, \hat{W}, d_{\{u\}})$ and $L_0, R_0 \leftarrow \text{Partition}(G, W_0, d_{\{u\}})$.
5. For $t = 0, 1, \ldots, \lceil 1/\epsilon \rceil$ do:
   (a) Let $v_1, \ldots, v_s$ be an ordering of S such that $f(v_1) \geq f(v_2) \geq \cdots \geq f(v_s)$, where $f(v) = d_{\hat{L}_t}(v) - d_{\hat{R}_t}(v)$. If $\max_j\{f(v_j) - f(v_{j+1})\} \geq D$, then let j be the first index for which $f(v_j) - f(v_{j+1}) \geq cD$, and return $\{v_1, \ldots, v_j\}$. Otherwise, continue.
   (b) Randomly select a set W_{t+1} from V' of $2l$ unchosen vertices.
   (c) $\hat{L}_{t+1}, \hat{R}_{t+1} \leftarrow \text{Partition}(G, \hat{W}, d_{L_t} - d_{R_t})$.
   (d) $L_{t+1}, R_{t+1} \leftarrow \text{Partition}(G, W_{t+1}, d_{L_t} - d_{R_t})$.

See Figure 3. The total number of vertices selected by procedure Find$_2$ is $s+s_1+s_2+2\hat{l}+(\lceil 1/\epsilon \rceil+1)\cdot 2l$. We assume that the bound $k \geq \Omega(\Delta^{-1-\epsilon}\sqrt{n\log n})$ is satisfied, and in that case, $\Delta^{-1-\epsilon} \leq O(k/\sqrt{n \log n}) \leq O(\sqrt{n/\log n}/m)$. Therefore $s \leq O(m^{1-2/(1+\epsilon)}n^{1/(1+\epsilon)}\log^{1-1/(1+\epsilon)} n) \leq o(n)$ and $\hat{l} \leq \hat{o}(n)$. Furthermore,

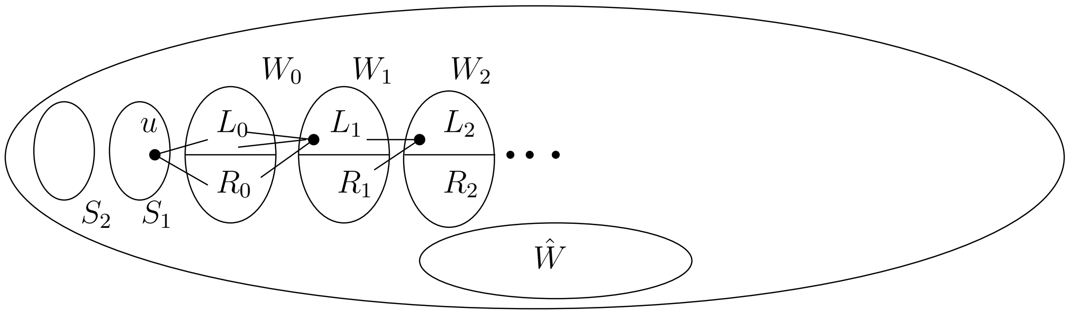

Figure 3: The sets created by procedure Find$_2$.

$m \leq n/k \leq O(\Delta^{1+\epsilon}\sqrt{n\log n}) \leq O(\sqrt{n\log n})$, so $s_1 \leq O(\sqrt{n}\log^{3/2} n)$. It follows that the total number of vertices selected by Find is at most $n/3$.

We assume w.l.o.g. that the vertex u chosen in Step 3 belongs to the cluster V_1. Let $K = \max_i |V_i|$. Recall that $A_i = |V_i|$.

Lemma 6.1. *If $k \geq \Omega(\sqrt{n\log n})$ then w.h.p., $A_1 \geq K - O(\Delta^{-1}\sqrt{n\log n})$.*

Proof. Let V_i be a cluster of maximum size. The probability that S_1 does not contain a vertex from V_i is at most $(1-a_i)^{s_1} \leq e^{-a_i s_1} \leq e^{-s_1/m} = 1/n^{\Omega(1)}$. Therefore, w.h.p., the set S_1 contains a vertex w from V_i. Denote $A'_j = |S_2 \cap V_j|$ for all j. By Corollary 2.5, w.h.p. $|d_{S_2}(v) - \mathrm{E}[d_{S_2}(v)]| \leq O(\sqrt{s_2\log s_2}) = O(\sqrt{n\log n})$ for every $v \in S_1$, where for $v \in V_j$ we have $\mathrm{E}[d_{S_2}(v)] = A'_j\Delta + s_2 r$. Thus, w.h.p.

$$A'_1\Delta + s_2 r + O(\sqrt{n\log n}) \geq d_{S_2}(u) \geq d_{S_2}(w) \geq A'_i\Delta + s_2 r - O(\sqrt{n\log n}),$$

so w.h.p. $A'_1 \geq A'_i - O(\Delta^{-1}\sqrt{n\log n})$. By Corollary 2.6, w.h.p., $|A'_j - \frac{A_j}{n}s_2| \leq O(\sqrt{a_j s_2\log n}) \leq O(\sqrt{n\log n})$ for every j. Therefore, w.h.p.

$$A_1 \geq \frac{n}{s_2}A'_1 - O(\frac{n}{s_2}\sqrt{n\log n}) \geq \frac{n}{s_2}A'_i - O(\Delta^{-1}\sqrt{n\log n}) \geq A_i - O(\Delta^{-1}\sqrt{n\log n}).$$

■

In the following lemma we use the same definitions of b_i^t, $\hat{b}_i^t$, r_i^t, $\hat{r}_i^t$, c_i^t, and t^* as in Section 5.

Lemma 6.2. *Suppose that $\max_i a_i < 2/3$, $k \geq \Omega(\Delta^{-1-\epsilon}\sqrt{n\log n})$ for some $\epsilon > 0$, $p, r \in [\frac{1}{4}, \frac{3}{4}]$, and $\Delta \leq O(1)$. Then, w.h.p.,*

1. *For all i and all $0 \leq t \leq t^*$, $|b_i^t - \hat{b}_i^t| \leq \hat{o}(a_1\Delta)$.*
2. *$b_1^0, \hat{b}_1^0 \geq (1-\hat{o}(1))\frac{2}{3}a_1\Delta$.*
3. *If $m \geq 3$, then for all $i > 1$,$-(1+\hat{o}(1)) \leq r_i^0, \hat{r}_i^0 \leq \hat{o}(1)$.*
4. *For all $1 \leq t \leq t^*$, $b_1^t, \hat{b}_1^t \geq \Omega(\Delta^{-\epsilon t}a_1\Delta)$.*

5. *If $m \geq 3$, then for all $1 \leq t \leq t^*$ and all $i > 1$, $-O(1) \leq r_i^t, \hat{r}_i^t \leq 1 - \Omega(1)$.*

Proof. Again, we prove items 2–5 only for b_i^t and r_i^t.

(1) By Theorem 4.1 we have w.h.p. that $|b_i^t - c_i^t| \leq d_i$ and $|\hat{b}_i^t - c_i^t| \leq d_i$ for all i and t, where

$$d_i \leq O(\sqrt{\frac{a_i}{l} \log n}(1 + \sqrt{a_i m})).$$

By Lemma 6.1 we have $A_1 \geq K - O(\Delta^{-1}\sqrt{n \log n}) \geq (1 - \hat{o}(1))K$. As $K \geq n/m$, we obtain that $a_1 \geq \Omega(1/m)$ and therefore $d_1 \leq O(\sqrt{\frac{a_1}{l}\log n} \cdot \sqrt{a_1 m}) \leq \hat{o}(a_1\Delta)$. Furthermore, for $i > 1$, $a_i \leq O(a_1)$ so $d_i \leq O(d_1) \leq \hat{o}(a_1\Delta)$.

Throughout this proof we assume that these inequalities hold. In particular, $|b_i^t - \hat{b}_i^t| \leq 2d_1 = \hat{o}(a_1\Delta)$.

(2) By Lemma 4.2, we have that

$$c_1^0 = 2a_1(1 - a_1)\Delta > \frac{2}{3}a_1\Delta$$

Therefore, $b_1^0 \geq c_1^0 - d_1 \geq (1 - \hat{o}(1))\frac{2}{3}a_1\Delta$.

(3) By Lemma 4.2, $c_i^0 = -2a_1 a_i \Delta$ for every $i > 0$. Therefore, $b_i^0 \leq c_i^0 + d_i \leq d_i \leq \hat{o}(a_1\Delta)$. Furthermore, $c_i^0 \geq -c_1^0$ (as $a_i \leq 1 - a_1$), so $b_i^0 \geq c_i^0 - d_i \geq -(1 + \hat{o}(1))c_1^0$. Thus, $-(1 + \hat{o}(1)) \leq r_i^0 \leq \hat{o}(1)$.

(4) We will show that $b_1^{t+1} \geq \Delta^{-\epsilon} b_1^t$ for all $t < t^*$. W.l.o.g. assume that $b_1^t \geq b_2^t \geq \cdots \geq b_m^t$.

As in the proof of item 4 in Lemma 5.1, we have that $|b_i^t - b_j^t| \leq \alpha/\Delta\sqrt{l}$ for all i, j where $\alpha = \hat{o}(1)$. (In this case we have that $|b_i^t - b_j^t| \leq |\hat{b}_i^t - \hat{b}_j^t| + \hat{o}(a_1\Delta) \leq O(\hat{b}_1^t - \hat{b}_2^t)$. Furthermore, we need to show that $V_1 \cap S \neq \phi$ and $S - V_1 \neq \phi$. Since $|V_1| \geq \Omega(n/m)$, we have that $|V_1 \cap S| \geq \Omega(\log n/\Delta^2)$. Moreover, as $|V - V_1| > n/3$, we have that $|(V - V_1) \cap S| \geq \Omega(m \log n/\Delta^2)$. In particular, we have that $V_1 \cap S \neq \phi$ and $S - V_1 \neq \phi$.)

Denote $F_0 = \gamma\Delta\sqrt{l}$ (where γ is defined in Lemma 4.3), $F = F_0 K/n$ and $Y = (\sum_{j=1}^m a_j b_j^t)/b_1^t$. By Lemma 4.3,

$$\begin{aligned} b_1^{t+1} &\geq F_0 a_1(1 - O(\Delta) - \frac{2}{9}\alpha^2)(b_1^t - \sum_{j=1}^m a_j b_j^t) - 3\frac{a_1}{\sqrt{l}} - d_1 \\ &\geq (1 - \hat{o}(1))(1 - Y)F b_1^t - 3\frac{a_1}{\sqrt{l}} - d_1 \end{aligned}$$

where the last inequality follows from the fact that $\Delta \leq O(1)$ and $A_1 \geq (1 - \hat{o}(1))K$. We have that $F_0 \geq \Omega(\Delta\sqrt{l}/m) \geq \Omega(1)$ and $b_1^t \geq \Omega(a_1\Delta)$. Therefore,

$$3\frac{a_1}{\sqrt{l}} \leq O(a_1 \frac{\Delta}{\sqrt{m \log n}}) \leq o(F_0 b_1^t)$$

and

$$d_1 \leq \hat{o}(a_1\Delta) \leq \hat{o}(F_0 b_1^t).$$

It follows that $b_1^{t+1} \geq ((1-\hat{o}(1))(1-Y) - \hat{o}(1))F_0 b_1^t$.

Now,

$$\sum_{j=1}^{m} a_j b_j^t \leq a_1 b_1^t + \sum_{j=2}^{m} a_j b_2^t = a_1 b_1^t + (1-a_1)b_2^t \leq \frac{2}{3}b_1^t + \frac{1}{3}b_2^t = (\frac{2}{3} + \frac{1}{3}r_2^t)b_1^t.$$

Therefore, $b_1^{t+1} \geq ((1-\hat{o}(1))\frac{1}{3}(1-r_2^t) - \hat{o}(1))F_0 b_1^t \geq \Omega(F_0 b_1^t) \geq \Omega(\Delta\sqrt{l}/m \cdot b_1^t) \geq \Delta^{-\epsilon} b_1^t$.

(5) By Lemma 4.2,

$$\begin{aligned}
b_i^{t+1} &\leq F_0 a_i (b_i^t - \sum_{j=1}^{m} a_j b_j^t + (O(\Delta) + \frac{2}{9}\alpha^2)(b_1^t - b_m^t)) + 3\frac{a_i}{\sqrt{l}} + d_i \\
&\leq F_0 a_i (b_i^t - \sum_{j=1}^{m} a_j b_j^t) + \hat{o}(F_0 a_i b_1^t) + \hat{o}(F b_1^t) \\
&\leq F \max(0, b_i^t - \sum_{j=1}^{m} a_j b_j^t) + \hat{o}(F b_1^t) \\
&= (\max(0, r_i^t - Y) + \hat{o}(1)) F b_1^t.
\end{aligned}$$

We have shown above that

$$b_1^{t+1} \geq ((1-\hat{o}(1))(1-Y) - \hat{o}(1))F b_1^t \geq (1-\hat{o}(1))(1-Y)F b_1^t,$$

where the last inequality follows as $1 - Y \geq \frac{1}{3}(1-r_2^t) \geq \Omega(1)$. Hence

$$r_i^{t+1} \leq \frac{\max(0, r_i^t - Y) + \hat{o}(1)}{(1-\hat{o}(1))(1-Y)} \leq (1+\hat{o}(1))\frac{\max(0, r_i^t - Y)}{1-Y} + \hat{o}(1).$$

As

$$\sum_{j=1}^{m} a_j b_j^t \geq \sum_{j=1}^{m} a_j b_m^t = b_m^t = r_m^t b_1^t,$$

we have that $Y \geq r_m^t$, and it follows that

$$r_i^{t+1} \leq (1+\hat{o}(1))\frac{r_i^t - r_m^t}{1 - r_m^t} + \hat{o}(1).$$

Using similar arguments we obtain that

$$\begin{aligned}
r_i^{t+1} &\geq (1+\hat{o}(1))\frac{\min(0, r_i^t - Y)}{1-Y} - \hat{o}(1) \geq (1+\hat{o}(1))\frac{r_i^t - 1}{1 - (\frac{2}{3} + \frac{1}{3}r_2^t)} - \hat{o}(1) \\
&= (1+\hat{o}(1)) \cdot 3\frac{r_i^t - 1}{1 - r_2^t} - \hat{o}(1).
\end{aligned}$$

From the lower and upper bounds on r_i^{t+1} and the fact that the number of iterations is constant we conclude that $-O(1) \leq r_i^t \leq 1 - \Omega(1)$ for all t. ■

Lemma 6.3. *Suppose that* $\max_i a_i < 2/3$, $k \geq \Omega(\Delta^{-1-\epsilon}\sqrt{n \log n})$, *and* $p, r \in [\frac{1}{4}, \frac{3}{4}]$. *Then, w.h.p., procedure* Find_2 *returns a subcluster of size* $\Omega(\log n/\Delta^2)$. *The running time of procedure* Find_2 *is* $O(m^4\Delta^{-4(1+\epsilon)}\log n + m^2\Delta^{-4}\log^3 n + m\Delta^{-1}n\log n)$.

Proof. Let α be the constant hidden in the requirement $\Delta \leq O(1)$ of Lemma 6.2. If $\Delta \leq \alpha$, then from Lemma 6.2 we conclude (similarly to the proof of Lemma 5.2) that w.h.p., procedure Find_2 returns a subcluster of size $\Omega(\log n/\Delta^2)$.

We now consider the case when $\Delta > \alpha$. W.l.o.g. assume that $\hat{b}^0_1 \geq \hat{b}^0_2 \geq \cdots \geq \hat{b}^0_m$. The proof of items 2 and 3 in Lemma 6.2 does not require that $\Delta \leq \alpha$. Therefore, we have that $\hat{b}^0_1 - \hat{b}^0_2 \geq \frac{1}{2}\hat{b}^0_1 \geq \Omega(\Delta/m)$. Thus, $\min_{v\in S\cap V_1} f(v) - \max_{v\in S-V_1} f(v) \geq \Delta\hat{l}(\hat{b}^0_1 - \hat{b}^0_2) - D \geq \Omega(\hat{l}/m) - D \geq D$, hence procedure Find_2 stops at Step 5a and $\{v_1, \ldots, v_j\} = V_1 \cap S$.

As the number of iteration is at most $\lceil 1/\epsilon \rceil = O(1)$, the time complexity of Find_2 is $O(s\hat{l} + l\hat{l} + s_1 s_2) = O(l\hat{l} + s_1 s_2) = O(m^4\Delta^{-4(1+\epsilon)}\log n + m^2\Delta^{-4}\log^3 n + mn\log n)$. ■

Combining Lemma 5.2 and the above lemma we conclude:

Theorem 6.4. *The above algorithm solves the clustering problem w.h.p. when* $k \geq \Omega(\Delta^{-1-\epsilon}\sqrt{n \log n})$ *for some* $\epsilon > 0$. *The running time of the algorithm is* $O(m^5\Delta^{-4(1+\epsilon)}\log n + m^3\Delta^{-4}\log^3 n + (m^2 + m\Delta^{-2})n\log n)$.

The requirement $k \geq \Omega(\Delta^{-1-\epsilon}\sqrt{n \log n})$ implies that $\Delta^{-1-\epsilon} \leq O(\sqrt{n/\log n}/m)$. Therefore,

$$m^5\Delta^{-4(1+\epsilon)}\log n \leq O(mn^2/\log n)$$
$$m^3\Delta^{-4}\log^3 n \leq O(m^{3-4/(1+\epsilon)}n^{2/(1+\epsilon)}\log^{3-2/(1+\epsilon)} n) \leq o(mn^2/\log n)$$

and

$$m\Delta^{-2}n\log n \leq O(m^{1-2/(1+\epsilon)}n^{1+1/(1+\epsilon)}\log^{1-1/(1+\epsilon)} n) \leq o(mn^2/\log n).$$

Furthermore, $m \leq O(\sqrt{n/\log n})$ so $m^2 n\log n \leq O(mn^{3/2}\sqrt{\log n}) \leq o(mn^2/\log n)$. Therefore, the time complexity of the algorithm is $O(mn^2/\log n)$.

Acknowledgements

R. Shamir was supported in part by the Israel Science Foundation (grant 309/02).

References

[1] A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns. *J. of Computational Biology*, 6:281–297, 1999.

[2] R. B. Boppana. Eigenvalues and graph bisection: An average-case analysis. In *Proc. 28th Symposium on Foundation of Computer Science (FOCS 87)*, pages 280–285, 1987.

[3] T. Carson and R. Impagliazzo. Hill-climbing finds random planted bisections. In *Proc. Twelfth Symposium on Discrete Algorithms (SODA 01)*, pages 903–909. ACM press, 2001.

[4] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Ann. Math. Statis.*, 23:493–507, 1952.

[5] A. E. Condon and R. M. Karp. Algorithms for graph partitioning on the planted partition model. *Random Structures and Algorithms*, 18(2):116–140, 2001.

[6] M. E. Dyer and A. M. Frieze. The solution of some random NP-hard problems in polynomial expected time. *J. of Algorithms*, 10(4):451–489, 1989.

[7] U. Feige and J. Kilian. Heuristics for semirandom graph problems. *J. of Computer and System Sciences*, 63(4):639–671, 2001.

[8] M. Jerrum and G. B. Sorkin. The Metropolis algorithm for graph bisection. *Discrete Applied Math*, 8:155–175, 1998.

[9] A. Jules. *Topics in black box optimization.* PhD thesis, U. California, 1996.

[10] F. McSherry. Spectral partitioning of random graphs. In *Proc. 42nd Symposium on Foundation of Computer Science (FOCS '01)*, pages 529–537, 2001.

[11] V. V. Petrov. *Sums of independent random variables.* Springer-Verlag, 1975.

[12] R. Shamir and D. Tsur. Improved algorithms for the random cluster graph model. In *Proc. 8th Scandinavian Workshop on Algorithm Theory (SWAT '02)*, LNCS 2368, pages 230–239. Springer-Verlag, 2002.

[13] P. van Beek. *Z. Wahrscheinlichkeitstheorie verw. Geb.*, 23(3):187–196, 1972.

[14] Z. Yakhini. Personal communications, 2000.

List of Symbols

Symbol	Meaning	Introduced on page
Δ	$p - r$	1
Γ	$\max_i \lvert \frac{\lvert V_i \rvert}{n} - \frac{1}{m} \rvert$	8
A_i	$\lvert V_i \rvert$	8
a_i	$\lvert V_i \rvert / n$	8
b_i^t	$\mathrm{I}(V_i, L_t, R_t)$	20
$c_i(f)$	$2a_i \sum_{j \neq i} a_j (p_{ij}^{>}(f) - p_{ij}^{<}(f))$	13
c_i^0	$c_i(d_{\{u\}})$	21
c_i^t	$c_i(d_{L_t} - d_{R_t})$	21
$d_S(v)$	Number of neighbors of v in S	3
$\mathrm{I}(V_i, L, R)$	$\frac{\lvert V_i \cap L \rvert - \lvert V_i \cap R \rvert}{\lvert L \rvert}$	12
K	$\max_i \lvert V_i \rvert$	26
k	$\min_i \lvert V_i \rvert$	2
m	Number of clusters	1
n	Number of vertices	2
p	Probability for an edge between vertices of same cluster	1
$p_{ij}^{>}(f)$	$\mathrm{P}\left[f(v) > f(w) \mid v \in V_i, w \in V_j\right]$	13
r	Probability for an edge between vertices of different clusters	1
r_i^t	b_i^t / b_1^t	20
V_i	i-th underlying cluster	1