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 $k = \min_i |V_i|$ 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

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

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

[‡]Department of Computer Science, Ben-Gurion University of the Negev. Email: `dekelts@cs.bgu.ac.il`

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. Our first result is an algorithm for the case when the sizes of the clusters are equal or almost equal:

Theorem 1.1. *There is an algorithm that solves the clustering problem with probability* $1 - O(1/n)$ *when* $\max_i |V_i|/k \le 1 + 1/(10000 \log n)$ *and* $k \ge 10^8 \Delta^{-1} \sqrt{n} \log n$. *The running time of the algorithm is* $O(m^4 \Delta^{-4} \log^3 n \cdot (m + \log n) + m\Delta^{-2} n \log n) = O((m/\log n + 1)n^2)$.

We also give an algorithm for the general case where the clusters have arbitrary sizes.

Theorem 1.2. *For every constant* $\epsilon > 0$ *there is an algorithm that solves the clustering problem with probability* $1 - O(1/n)$ *when* $k \ge 10^6 2^{8 \cdot 3^{\lceil 1/\epsilon \rceil + 1}} \Delta^{-1} \sqrt{n \log n}(\sqrt{\log n} + \Delta^{-\epsilon})$. *The running time of the algorithm is* $O(m^5 \Delta^{-4(1+\epsilon)} \log n + m^3 \Delta^{-4} \log^3 n + (m^2 + m\Delta^{-2}) n \log n) = O(mn^2/\log n)$.

We note that we did not optimize the constants in the bounds on k in the two theorems since optimizing these constants would have complicated the proofs.

1.1 Previous Results

The random graph model was studied by several authors [1–6, 8, 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 reduced to $O(n \log n)$ [14].

Independently, McSherry [11] gave a polynomial time 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 $\Delta = \omega(n^{-1/2} \log^3 n)$, while

(a) General case.

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

(b) Equal sized clusters.

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

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 $k \le n/m$. 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.

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

1.2 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 [4] (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'.

We will say that each pair v, w in step 1 are in a *duel*, and the *winner* of the duel is the vertex whose expression is larger.

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, and the *secondary imbalance of* L, R is the second largest value. 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, sorted in a non increasing order (namely, $l_1 \geq l_2 \geq \cdots \geq l_m$). 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 there is a duel in which one vertex is from V_1 and the other vertex is not. Furthermore, for such a duel, it is more likely that the vertex from V_1 will win (namely, the vertex 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. 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 win (which is equal to the probability that $X > 0$ plus half of the probability that $X = 0$) is $1/2 + \Omega(\min(1, (l_i - l_j)\Delta/\sqrt{|L|}))$ 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 value of l_2 much smaller than l_1 (for example, $l_2 \leq \frac{1}{10}l_1$), 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.

In order to build the sets L and R with the desired imbalances, our algorithm performs the following steps:

Initialization Build sets L_0 and R_0 with "large" imbalance and "small" secondary imbalance.

Imbalance amplification For $t = 1, 2, \ldots$, build a pair of sets L_t, R_t from L_{t-1}, R_{t-1} such that the imbalance of L_t, R_t is at least twice the imbalance of L_{t-1}, R_{t-1}, and the secondary imbalance of L_t, R_t is much smaller than the imbalance of L_t, R_t.

For building the pair L_t, R_t from L_{t-1}, R_{t-1}, we use a process similar to the CK algorithm. We perform *duels* between disjoint pairs of vertices from $V - (L_{t-1} \cup R_{t-1})$: For each pair v, w we compute the expressions $d_{L_{t-1}}(v) - d_{R_{t-1}}(v)$ and $d_{L_{t-1}}(w) - d_{R_{t-1}}(w)$. The vertex whose expression is larger is added to L_t, and the other vertex is added to R_t. Our algorithm and its analysis are similar to the CK algorithm, but there are several important differences:

- As described above, the analysis of the CK algorithm involves only l_1, while our analysis is for all the l_i-s (in our algorithm, the l_i-s are the L_t, R_t-imbalances for the current pair L_t, R_t).

- In the CK algorithm, the outcome of a duel depends on the outcome of all previous duels. Due to the strong dependency between the duels, the analysis of the CK algorithm uses bounds on biased random walk. In our algorithm, the duels that are performed when building L_t and R_t depend only on the outcome of the duels that were performed when building L_{t-1} and R_{t-1}. Due to this limited dependency, we do not need to use random walks in our analysis, and we are able to give tighter bounds on behavior of the l_i-s.

- In our algorithm, the behavior of the l_i-s goes a phase transition when t increases: For small t, the l_i-s behave "nicely", that is $l_2 \leq \frac{1}{10}l_1$. However, for large t, the behavior changes, and in particular l_2 might be very close to l_1. We therefore need to stop the imbalance amplification stage at the correct step. In contrast, the CK algorithm does not require stopping at a precise step.

- Since in our algorithm we have 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).

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 may lead to improved heuristic algorithms.

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. When we need to distinguish between several graphs, we will use n_G to denote the number of vertices in the graph G, and we will do the same for other parameters of G.

For proving the correctness of our algorithms we need the following known results which give estimates on the sum of independent random variables.

Theorem 2.1 (Esseen's Inequality)**.** *Let $X_1, \ldots, X_n$ be independent random variables such that $E\left[|X_i|^3\right] < \infty$ for $i = 1, \ldots, n$. Let $X = \sum_{i=1}^n X_i$, $B_n = \sum_{i=1}^n E\left[X_i^2\right]$, and $L_n = B_n^{-3/2} \sum_{i=1}^n E\left[|X_i - E[X_i]|^3\right]$. Then $\left|P[X > 0] - \Phi(E[X]/\sqrt{B_n})\right| \leq AL_n$, 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 [12, p. 111]. Esseen's inequality holds for $A = \frac{4}{5}$ [13].

Esseen's inequality gives an estimate on the distribution of a sum of independent random variables. A stronger estimate on the tail of this distribution and other distributions is given by the following theorem of McDiarmid [10] (this theorem generalizes previous results by Hoeffding [7] and others).

Theorem 2.2. *Let $X_1, \ldots, X_n$ be random variables, with $0 \leq X_i \leq 1$ for all i, and let $Y_i = \sum_{j=1}^i X_j$. If for all $i \geq 2$, $E[X_i|Y_{i-1} = x]$ is a non-increasing function of x, then $P[Y_n - E[Y_n] \geq a] \leq L(n, E[Y_n], a)$ for every $a \leq n - E[Y_n]$, where $L(n, \mu, a) = \left(\frac{\mu}{\mu+a}\right)^{\mu+a} \left(\frac{n-\mu}{n-\mu-a}\right)^{n-\mu-a}$.*

We shall use Theorem 2.2 on two types of distributions: sum of independent random variables distribution and the hypergeometric distribution (see [10]). Using standard bounds on $L(n, \mu, a)$ we obtain the following corollaries:

Corollary 2.3. *Let $X_1, \ldots, X_n$ be independent Bernoulli random variables, and let $X = \sum_{i=1}^n X_i$. If $E[X] \leq y$ then $P[|X - E[X]| > a] \leq 2e^{-a^2/3y}$ for every $0 \leq a \leq y$.*

Corollary 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 of size s. If $\frac{k}{n}s \leq y$ then $P\left[\left||B \cap S| - \frac{k}{n}s\right| > a\right] \leq 2e^{-a^2/3y}$ for every $0 \leq a \leq y$.*

In the next sections, we use Corollary 2.4 in a slightly different scenario: The set S is chosen randomly from $A - S'$, where S' is a random set subset of A that is chosen before S is chosen. Choosing $S' \subseteq A$ and then $S \subseteq A - S'$ is equivalent to choosing $S \subseteq A$ and then $S' \subseteq A - S$. Therefore, we can still use Corollary 2.4 on S.

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 $A_i = |V_i|$ and $a_i = |V_i|/n$. We also denote $A_{\max} = \max_i A_i$ and $a_{\max} = A_{\max}/n$. 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 Find(G, S, Δ_e) that receives a random cluster graph $G = (V, E)$, a set $S \subseteq V$ and a lower bound Δ_e on Δ. Find(G, S, Δ_e) returns a subcluster of G of size $\Omega(\log n/\Delta_e^2)$ that is contained in S. To compute this subcluster, procedure Find only considers vertices and edges in G_S. Now, we use the following algorithm for solving the clustering problem: Repeatedly, find a subcluster S in the input 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. We now give a formal description of the algorithm. The input to algorithm Solve is an input graph G, and a lower bound Δ_e on the value of Δ. In the following, when we write $t - 1$, the subtraction is performed modulo 3, that is $t - 1 = 3$ when $t = 1$.

```
    Solve($G, \Delta_e$):
1:  while True
2:     Randomly partition the vertices of $G$ into equal sized sets $W_1$, $W_2$, and $W_3$.
3:     $S_0 \leftarrow$ Find$(G, W_3, \Delta_e)$.
4:     Let $n$ be the number of vertices of $G$, $s = 200\log n/\Delta_e^2$ and $D = 4\sqrt{3}\sqrt{s\log n}$.
5:     For $t = 1, 2, 3$ do
6:        If $t = 1$ then let $S'_{t-1}$ be a random subset of $S$ of size $s$.
7:           else let $S'_{t-1}$ be a random subset of $S_{t-1}$ of size $s$.
8:        Let $v_1^t, \ldots, v_{n/3}^t$ be an ordering of $W_t$ such that $d_{S'_{t-1}}(v_1^t) \geq d_{S'_{t-1}}(v_2^t) \geq \cdots \geq d_{S'_{t-1}}(v_{n/3}^t)$.
9:        If $\max_j\{d_{S'_{t-1}}(v_j^t) - d_{S'_{t-1}}(v_{j+1}^t)\} \leq D$ then output $V$ and stop.
10:       Let $j$ be the index such that $d_{S'_{t-1}}(v_j^t) - d_{S'_{t-1}}(v_{j+1}^t)$ is maximum.
11:       Let $S_t = \{v_1^t, \ldots, v_j^t\}$.
12:       If $|S_t| < s$ then stop.
13:    Output $S_1 \cup S_2 \cup S_3$ and delete the vertices in $S_1 \cup S_2 \cup S_3$ from $G$.
```

See also Figure 1. For simplicity, we omit floors and ceilings from the description of the algorithm and its proof.

Note that the partition of the graph into three parts is done in order to avoid dependencies. This partition guarantees that on step 8 of the algorithm when $t = 1$,

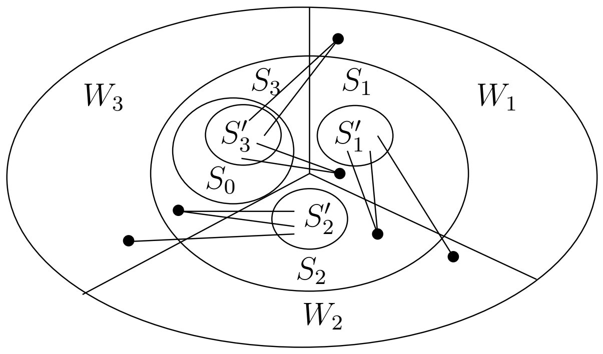

Figure 1: The sets created by algorithm Solve. The edges represented are counted by the algorithm. For example, the set S_1 is built by looking at the edges between W_1 and S'_3.

for every $u \in S'_3$ and $v \in W_1$, the existence of an edge between u and v is independent of the choice of the set S'_3. 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_3, the existence of the edges between W_3 and W_1 has not been determined when the algorithm enters step 8, so for each $u \in W_3$ and $v \in W_1$, the event that there is an edge between u and v is independent of the choice of S'_3. In fact, this would remain true, even if the set S'_3 is chosen by an adversary that can only look at edges in G_{W_3}. We note that it is suffices to partition the input graph G into two parts. However, we chose to use three parts to simplify the presentation.

The above principle will be used in other parts of our algorithms: We say that a pair of vertices (u, v) is *considered* in some step of the algorithm if in that step, the existence of an edge (u, v) is checked by the algorithm (for example, (u, v) is considered when the value of $d_S(u)$ is computed for some set S that contains v). A main design principle in our algorithms is to try to consider each pair (u, v) only once during the run of the algorithm.

Unfortunately, we cannot meet the goal above: During algorithm Solve, a pair (u, v) is considered only once in each iteration of the algorithm, but can be considered several times in different iterations. Thus, we need to change our goal of considering each pair only once. We assume that procedure Find consists of two stages. The first stage determines the cluster V_i from which the procedure will return a subcluster, and the second stage finds a subcluster of V_i. We say that a pair of vertices (u, v) is *important* if it is considered during the first stage of some call to procedure Find. We will design our algorithms so they will have the following property:

Property 1. Each important pair is considered only once during the entire run of the algorithm.

This property suffices for avoiding dependencies between the iterations of algorithm Solve as the remaining graph G in the i-th iteration of the algorithm depends only on the important pairs of the first $i - 1$ calls to procedure Find (assuming that the first $i - 1$ iterations successfully found $i - 1$ clusters of the graph). In other words, the unimportant pairs that were considered during the first $i-1$ iterations (in the second stage of procedure Find or in steps 5–12 of algorithm Solve) have no influence on the i-th iteration, so we can assume for the analysis that there were no unimportant edges in the first $i-1$ iterations.

Let $t(G)$ be an upper bound on the running time of procedure Find on an input graph G. The bounds we will use below have the property that $t(G') \leq t(G)$ for every every cluster collection G' of G (this follows from the fact that $t(G)$ will be a polynomial in m_G, $\log n_G$, and Δ_G^{-1}, and we have that $m_{G'} \leq m_G$, $n_{G'} \leq n_G$, and $\Delta_{G'} = \Delta_G$). We use this property in the proof of the following lemma.

Lemma 3.1. Suppose that $\text{Find}(G, S, \Delta_e)$ returns a subcluster of size at least $1200 \log n/\Delta_e^2$ with probability $1 - O(1/n^3)$. If $k \geq \sqrt{n}$ and $\Delta_e \leq \Delta$ then algorithm Solve solves the clustering problem with probability $1-O(1/n)$. The running time of algorithm Solve is $O(mt(G) + mn\log n/\Delta_e^2)$ with probability $1 - O(1/n)$.

Proof. The main idea for the proof is that the number of edges between a vertex v and a set $S \subseteq V_i$ is a random variable whose distribution depends on whether $v \in V_i$. With high probability, the values $d_S(v)$ for all $v \in V_i$ are significantly larger than the values $d_S(v)$ for all $v \notin V_i$, so the algorithm correctly separates the vertices of V_i from the rest of the vertices.

Consider the first iteration of the algorithm. Assuming that $\text{Find}(G, W_3, \Delta_e)$ did not fail, let V_i be the cluster for which $S_0 \subseteq V_i$. We shall show that if G consists of one cluster, then with probability $1 - O(1/n^3)$ the algorithm stops at Step 9, and otherwise, $S_1 \cup S_2 \cup S_3 = V_i$ with probability $1 - O(1/n^3)$.

We claim that the first iteration fails only if at least one of the following events happens:

1. $\left|d_{S'_{t-1}}(v) - E\left[d_{S'_{t-1}}(v)\right]\right| > \frac{1}{2}D$ for some $t \in \{1,2,3\}$ and some vertex $v \in W_t$ (assuming that the set S'_{t-1} was built by algorithm Solve).

2. $|V_j \cap W_t| < \frac{1}{6}A_j$ for some j and t.

3. $\text{Find}(G, W_3, \Delta_e)$ failed.

We will first bound the probability of these events, and then we will show that if none of these events happens, then the first iteration does not fail.

To bound the probability of the first event, note that $E\left[d_{S'_{t-1}}(v)\right] \leq s$, $2\sqrt{3}\sqrt{s\log n} \leq s$, and the edges between W_{t-1} and W_t are independent of the choice of S. Therefore, using Corollary 2.3 we get that for a fixed t and a fixed $v \in W_t$, the probability that $\left|d_{S'_{t-1}}(v) - E\left[d_{S'_{t-1}}(v)\right]\right| > 2\sqrt{3}\sqrt{s\log n} = \frac{1}{2}D$ is $O(e^{-(2\sqrt{3}\sqrt{s\log n})^2/3s}) = O(n^{-4})$. Thus, event 1 happens with probability $O(n^{-3})$.

For fixed j and t, Corollary 2.4 gives that the probability that $||V_j \cap W_t| - \frac{1}{3}A_j| > \frac{1}{3}A_j$ is $O(e^{-(\frac{1}{3}A_j)/3}) = O(e^{-\frac{1}{9}\sqrt{n}})$. Therefore, event 2 happens with probability $O(3m \cdot e^{-\frac{1}{9}\sqrt{n}}) = o(n^{-3})$. Finally, event 3 happens with probability $O(n^{-3})$.

Now assume that events 1–3 do not happen, and consider the first iteration of steps 5–12 (namely, $t = 1$). For $v \in V_i$ we have that $E\left[d_{S'_3}(v)\right] = sp$, and for $v \notin V_i$ we have $E\left[d_{S'_3}(v)\right] = sr$. Using the assumption that event 1 does not happen, we have that if G consists of one cluster, then $|d_{S'_3}(v) - d_{S'_3}(v')| \leq D$ for all $v, v' \in W_1$, and therefore, algorithm Solve stops at Step 9. Otherwise, for two vertices $v, v' \in W_1$, if either $v, v' \in V_i$ or $v, v' \notin V_i$ then $|d_{S'_3}(v) - d_{S'_3}(v')| \leq D$, and if $v \in V_i$ and $v' \notin V_i$ then $d_{S'_3}(v) - d_{S'_3}(v') \geq sp - sr - D \geq s\Delta_e - D > D$. It follows that algorithm Solve does not stop at Step 9 when $t = 1$, and furthermore, $S_1 = V_i \cap W_1$. Moreover, since event 2 does not happen, it follows that $|V_i \cap W_1| \geq \frac{1}{6}A_i \geq \frac{1}{6}|S_0| \geq s$, so the algorithm does not stop at Step 12 when $t = 1$. Using the same arguments, $S_2 = V_i \cap W_2$ and $S_3 = V_i \cap W_3$.

Now, using the same arguments as above, the failure probability in the i-th iteration, assuming the first $i-1$ iterations succeeded, is $O(1/n_i^3)$, where n_i is number of vertices in the graph remaining after the first $i-1$ iterations. We have that $n_i \geq k \geq \sqrt{n}$, so the failure probability at each iteration is $O(1/n^{3/2})$. It follows that the failure probability of the algorithm is $O(m/n^{3/2})$, and since $m \leq n/k \leq \sqrt{n}$, we obtain that the overall failure probability is $O(1/n)$.

From the correctness of the algorithm, we have that algorithm Solve performs exactly m iterations with probability $1 - O(1/n)$ (note that if events 1–3 occur during the run of the algorithm, the set $S_1 \cup S_2 \cup S_3$ which is removed from the graph may not be a cluster, and therefore the algorithm may perform more than m iterations). The time complexity of a single iteration is $O(t(G) + n \log n/\Delta_e^2)$, where the second term is the time for computing the degrees at step 8 (sorting the degrees can be done in $O(n)$ time by bin sorting). ■

Algorithm Solve requires the knowledge of some lower bound Δ_e on Δ. Such bound can be obtained from the bounds on k in Theorems 1.1 and 1.2. For example, if the bound on k in Theorem 1.1 is satisfied, then $\Delta \geq 10^8 \log n/\sqrt{n}$. To simplify the presentation, we will assume for the next algorithms that Δ and m are known. The case where these parameters are not known can be handled by replacing the terms Δ and m in the algorithms descriptions by Δ_e and m_e, where Δ_e is a lower bound on Δ and m_e is an upper bound on m. It is easy to verify that the following proofs remain correct under these replacements. The intuition for this is that when using Δ_e and m_e, the algorithm builds larger sets, and therefore the error due to "randomness" is smaller.

3.1 Handling large clusters and close edge probabilities

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

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

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

(R3) $a_{\max} < \frac{2}{3}$ or $m = 1$.

Note that if a graph G satisfies (R1) then 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.

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. That is, we will show that if we have a procedure Find that works on graphs that satisfy (R1)–(R3), then we can solve the clustering problem on every graph that satisfies (R1).

Let Solve_1 be a procedure that runs the first iteration of algorithm Solve on its input (namely, Solve_1 finds one cluster in the input graph). 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 as follows:

```
    $\text{Solve}_2(G)$:
1:  Randomly partition the vertices of $G$ into equal sized sets $W_1$, $W_2$, and $W_3$.
2:  Let $s = 16000 \log n/\Delta^2$ and $D = 8\sqrt{3}\sqrt{s \log n}$.
3:  Randomly choose a set $W_3'$ of unmarked vertices from $W_3$ of size $s$, and mark all
    the vertices of $W_3'$.
4:  Let $v_1, \dots, v_{n/3}$ be an ordering of $W_1$ such that $d_{W_3'}(v_1) \geq d_{W_3'}(v_2) \geq \cdots \geq$
    $d_{W_3'}(v_{n/3})$.
5:  **If** $\max_j\{d_{W_3'}(v_j) - d_{W_3'}(v_{j+1})\} \leq D$
6:      $R \leftarrow \text{Solve}_1(G)$.
7:      **Output** $R$.
8:      **If** $R = V$ **then stop**.
9:                  **else call** $\text{Solve}_2(G_{V-R})$.
10: **else**
11:     Let $j$ be the first index for which $d_{W_3'}(v_j) - d_{W_3'}(v_{j+1}) > D$.
12:     Let $S_1 = \{v_1, \dots, v_j\}$ and $d_0 = d_{W_3'}(v_j)$.
13:     **For** $t = 2, 3$ **do**
14:         Randomly choose a set $W_{t-1}'$ of unmarked vertices from $W_{t-1}$ of size $s$, and
            mark all the vertices of $W_{t-1}'$.
15:         Let $S_t = \{v \in W_t | d_{W_{t-1}'}(v) \geq d_0 - D/2\}$.
16:     $R \leftarrow S_1 \cup S_2 \cup S_3$.
17:     **Call** $\text{Solve}_2(G_R)$ and $\text{Solve}_2(G_{V-R})$.
```

Note that the marks on the vertices are maintained globally, namely, a vertex that is marked when Solve_2 is called on a graph G remains marked when Solve_2 is called on subgraphs of G. In order to avoid dependencies, we can make sure that the vertices that are chosen by Solve_2 in steps 3 and 14 will not be chosen in other steps of the algorithm, and therefore every pairs of vertices that is considered by Solve_2 in steps 4 and 15 is considered only once during the entire algorithm.

Lemma 3.2. If procedure Find returns with probability $1-O(1/n^3)$ a subcluster of size at least $1200 \log n/\Delta^2$ on graphs that satisfy (R1)–(R3), then with probability $1 - O(1/n)$, algorithm Solve_2 solves the clustering problem on graphs that satisfy (R1) and (R2). The running time of algorithm Solve_2 is $O(mt(G) + mn \log n/\Delta^2)$ with probability $1 - O(1/n)$.

Proof. The lemma will be proved using the fact that the number of edges between a vertex v and a random set of vertices T is a random variable, whose distribution depends on the size of the cluster V_i that contains v. In particular, if V_i is a cluster of size at least $\frac{2}{3}n$, then w.h.p., the vertices of V_i will have more edges to T than vertices in $V - V_i$, and this allows the algorithm to split the graph G into two cluster collections, G_{V_i} and G_{V-V_i}, in steps 11–17. The algorithm might execute steps 11–17 even if $a_{\max} < \frac{2}{3}$. Therefore, we need to show that in that case, G_R and G_{V-R} are cluster collections.

The following events may cause algorithm Solve$_2$ to fail during the first invocation of the algorithm:

1. $\left|d_{W'_{t-1}}(v) - E\left[d_{W'_{t-1}}(v)\right]\right| > \frac{1}{4}D$ for some $t \in \{1, 2, 3\}$ and some vertex $v \in W_t$, (as before, $t - 1$ is computed modulo 3).
2. $||V_i \cap S| - a_i s| > \frac{1}{2}D$ for some i.
3. Solve$_1(G)$ failed (assuming step 6 is executed).

We claim that events 1–3 happen with probability $O(n^{-3})$: For event 1 this is true due to Corollary 2.3. For event 2, this follows from Corollary 2.4. Finally, the proof of Lemma 3.1 gives that probability that event 3 happens is $O(n^{-3})$, assuming that (R3) is satisfied.

Now, assume that events 1 and 2 do not happen. We first show that (R3) is satisfied if step 6 is executed. To show this, we will show that if (R3) is not satisfied, then the condition in step 5 is not satisfied. Suppose that (R3) is not satisfied, namely $a_i \geq 2/3$ for some cluster V_i. Denote $A'_j = |V_j \cap W'_3|$. For $v \in V_j \cap W_1$ we have that $E\left[d_{W'_3}(v)|A'_j\right] = A'_j p + (s - A'_j)r = sr + A'_j\Delta$. Therefore, for $v \in V_i \cap W_1$ and $v' \in V_j \cap W_1$ (for $j \neq i$) we have

$$\begin{aligned} d_{W'_3}(v) - d_{W'_3}(v') &> (sr + (a_i s - \frac{1}{2}D)\Delta - \frac{1}{2}D) - (sr + a_j s\Delta + \frac{1}{2}D\Delta + \frac{1}{2}D) \\ &= (a_i - a_j)s\Delta - (1 + \Delta)D \geq \frac{1}{3}s\Delta - 2D \geq D. \end{aligned}$$

Since this is true for all v and v', we obtain that $\min_{v\in V_i\cap W_1} d_{W'_3}(v) - \max_{v\in W_1 - V_i} d_{W'_3}(v) > D$, so the condition in step 5 is not satisfied. For the rest of the proof, assume that events 1–3 do not happen.

If the condition in step 5 of the algorithm is satisfied then R is a cluster and G_{V-R} is a cluster collection. We next show that if the condition in step 5 is not satisfied then G_R and G_{V-R} are cluster collections. To show this, we will show that G_{S_t} is a cluster collection of G_{W_t} for $t = 1, 2, 3$, and moreover, the collections G_{S_1}, G_{S_2}, and G_{S_3} contain the same clusters, namely $V_i \cap W_1 \subseteq S_1$ if and only if $V_i \cap W_2 \subseteq S_2$, and if and only if $V_i \cap W_3 \subseteq S_3$.

Suppose that the condition in step 5 is not satisfied, namely $\max_j\{d_{W'_3}(v_j) - d_{W'_3}(v_{j+1})\} > D$. Then, for every cluster V_i, the set $V_i \cap W_1$ is contained in either S_1 or $W_1 - S_1$ (as for two vertices v and v' from the same cluster we have $|d_{W'_3}(v) - d_{W'_3}(v')| \leq \frac{1}{2}D$). In other words, G_{S_1} is a cluster collection of G_{W_1}.

For two vertices $v \in W_1$ and $v' \in W_2$ from some cluster V_i, we have that $E\left[d_{W'_3}(v)\right] = E\left[d_{W'_1}(v')\right]$. Therefore, $|d_{W'_3}(v) - d_{W'_1}(v')| \leq \frac{1}{2}D$. Furthermore, if $V_i \cap W_1 \subseteq S_1$ then $d_{W'_3}(v) \geq d_0$ and otherwise $d_{W'_3}(v) < d_0 - D$. In the former case we have $d_{W'_1}(v') \geq d_0 - \frac{1}{2}D$, and in the latter case $d_{W'_1}(v') < d_0 - \frac{1}{2}D$. We conclude that $V_i \cap W_2$ is contained in either S_2 or $W_2 - S_2$, and $V_i \cap W_2 \subseteq S_2$ if and only if $V_i \cap W_1 \subseteq S_1$. The same can be shown for $V_i \cap W_3$. Therefore, G_R and G_{V-R} are cluster collections.

Let $G_1, \dots, G_{m'}$ denote the graphs on which procedure Solve_1 is called during the run of $\mathrm{Solve}_2(G)$. From the above, we have that the failure probability of the algorithm is $O(\sum_{i=1}^{m'} n_{G_i}^{-3}) = O(m'/n^{3/2}) = O(1/n)$. We now compute the running time of algorithm Solve_2. The total number of calls to Solve_2 is at most $2m$ (there are at most m calls to Solve_2 in which the condition in step 5 is satisfied, and at most m calls in which the condition is not satisfied). Therefore, the total time not including the calls to Solve_1 is $O(m \cdot sn) = O(mn \log n/\Delta^2)$. The total time of the calls to Solve_1 is $O(\sum_{i=1}^{m'}(t(G_i) + n_{G_i} \log n_{G_i}/\Delta^2)) = O(mt(G) + mn \log n/\Delta^2)$. ■

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

```
   Solve$_3$($G$):
1: Build a graph $G'$ on the vertex set of $G$ in the following way: For every pair of
   vertices $u$ and $v$, 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: **Call** Solve$_2$($G'$).
```

Lemma 3.3. If procedure Find returns with probability $1-O(1/n^3)$ a subcluster of size at least $1200 \log n/\Delta^2$ on graphs that satisfy (R1)–(R3) (where (R1) is satisfied with constants α, β, and c), then with probability $1 - O(1/n)$, algorithm Solve_3 solves the clustering problem on in input graphs that satisfy (R1) (with constants α, β, and $2^\beta \cdot c$). The running time of Solve_3 is $O(mt(G') + mn \log n/\Delta^2)$ with probability $1 - O(1/n)$.

Proof. Clearly, G' is a random cluster graph on the clusters $V_1, \dots, V_m$ with edge probabilities $p_{G'} = \frac{3}{4}p_G + \frac{1}{4}(1-p_G) = \frac{1}{2}p_G + \frac{1}{4}$ and $r_{G'} = \frac{1}{2}r_G + \frac{1}{4}$. Moreover, G' satisfies (R1) (as $\Delta_{G'} = \frac{1}{2}\Delta_G$) and (R2), so by Lemma 3.2, algorithm $\mathrm{Solve}_3(G)$ solves the clustering problem with probability $1 - O(1/n)$. ■

4 The partition procedure

In the previous section, we presented the basic structure of our algorithms. We still need to show how to build procedure Find, namely, how to find a subcluster of size at least $1200 \log n/\Delta^2$. 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 [4]. 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, with $|L| = |R|$, 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). A pair of sets L, R for which the secondary imbalance of L, R is at most δ times the imbalance of L, R will be called an *δ-unbalanced pair*. We will show later that given a δ-unbalanced pair (for some constant $\delta < 1$) whose imbalance is large enough, it is possible to generate a subcluster. Therefore, our goal is to generate such a pair.

Let T be a random subset of V and let $f : T \to \mathbb{N}$ 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, $f(u)$ and $f(v)$ have the same distribution. For example, if u is some vertex of G and T is a random subset of $V - \{u\}$ then $f(v) = d_{\{u\}}(v)$ 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 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 of the other clusters, and the variance of $f(v)$ is "small" for all v. We can use this property of f to build an unbalanced pair L, R by performing a series of *duels* between random pairs of vertices of the graph. In a duel between the vertices v and w, we compute the values $f(v)$ and $f(w)$ and the *winner* is the vertex whose f-value is larger. If f has the property described above then a vertex from V_1 is likely to win in a duel against a vertex from $V - V_1$, so by taking L to be the set of winners and R to be the set of losers, we obtain that the L, R-imbalance of V_1 is large.

Formally, we define procedure Partition as follows (here T is a set of vertices):

```
   Partition($G, T, f$):
1: Begin with two empty sets $L$ and $R$. 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 [4], and the cluster function of Lemma 4.3 is also used in [4].

We now analyze procedure Partition. We begin with some definitions. Let $p^{>}_{ij}(f) = P\left[f(v) > f(w) | v \in V_i \cap T, w \in V_j \cap T\right]$ (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 and R are the output of a call to Partition(G, T, f) and denote $b_i = \mathrm{I}(V_i, L, R)$.

Before we analyze procedure Partition, consider a process similar to procedure Partition which perform duels between pairs of vertices that are randomly chosen from all vertices of the graph, *with repetitions* (we also assume that $f(v)$ is defined for all vertices of the graph). Let L' (resp., R') be the set of winners (resp., losers), and let $b'_i = \mathrm{I}(V_i, L', R')$. It is easy to verify that $E\left[b'_i\right] = c_i(f)$. Moreover, Corollary 2.3

gives that $|b'_i - E[b'_i]| = O(\sqrt{a_i l^{-1} \cdot \log n})$ with high probability. This analysis uses the fact that the vertices chosen by the process are independent. However, in procedure Partition, there are dependencies between the vertices of the different pairs. We will show below that these dependencies do not change the picture by much. That is, b_i behaves similarly to b'_i.

Theorem 4.1. *Let T is a random subset of V of size $2l$. If $k \geq 1200 \log n$, $l \geq 9 \log n / a_{\max}$ and $l \leq n/4$, then with probability $1 - O(1/n^{3.5})$, $|b_i - c_i(f)| \leq 72 a_{\max} \cdot \sqrt{ml^{-1} \log n}$ for all i.*

Proof. Since T is a random set, the process of choosing T and then randomly 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. 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 bound each of the values $I_i^1, \ldots, I_i^4$ to obtain a bound on b_i. We will now give an estimate on I_i^1 for some fixed i. We will bound I_i^1 from above and below with random variables that are sums of independent random variables.

Denote $a_j^r = P[u_r \in V_j]$ and $S = \{u_1, \ldots, u_{r-1}\}$. Clearly, $a_j^r = |V_j - S|/|V - S|$. Applying Corollary 2.4 (with $y = A_j$), we have with probability $1 - O(mn \cdot n^{-6}) = 1 - O(1/n^4)$ that $||V_j \cap S| - (r-1)a_j| \leq \sqrt{3 \cdot 6 \cdot A_j \log n}$ for all j and r (we use the fact that $A_j \geq k \geq 1200 \log n$ and thus $\sqrt{18 A_j \log n} \leq A_j$). For the rest of the proof assume that these inequalities hold for all j and r. Therefore,

$$\begin{aligned}|a_j - a_j^r| &= \left|\frac{|V_j|}{n} - \frac{|V_j - S|}{n-(r-1)}\right| = \left|\frac{|V_j| - |V_j - S|}{n-(r-1)} - \frac{(r-1)|V_j|}{n(n-(r-1))}\right| \\ &= \frac{1}{n-(r-1)} \left||V_j \cap S| - (r-1)a_j\right| \leq \frac{2}{n}\sqrt{18 A_j \log n} = 6\sqrt{2} \cdot \sqrt{a_j n^{-1} \log n}\end{aligned}$$

for all j and r.

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,

$$\begin{aligned}P[X_t = 1] &= a_i^{2t-1} \sum_{j \neq i} a_j^{2t} p_j \\ &= a_i \sum_{j \neq i} a_j p_j + a_i \sum_{j \neq i} (a_j^{2t} - a_j) p_j + (a_i^{2t-1} - a_i) \sum_{j \neq i} a_j^{2t} p_j \\ &\leq a_i \sum_{j \neq i} a_j p_j + a_i \sum_{j=1}^{m} |a_j^{2t} - a_j| + |a_i^{2t-1} - a_i| \\ &\leq a_i \sum_{j \neq i} a_j p_j + a_i \sum_{j=1}^{m} 6\sqrt{2}\sqrt{a_j n^{-1} \log n} + 6\sqrt{2}\sqrt{a_i n^{-1} \log n}.\end{aligned}$$

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,

$$P[X_t = 1] \le a_i \sum_{j \ne i} a_j p_j + 6\sqrt{2}\sqrt{a_i n^{-1} \log n}(1 + \sqrt{a_i m}).$$

Define the quantity in the last equation by θ. As $P[X_t = 1] \le \theta$ 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 $k \ge 1200 \log n$, we have that $\sqrt{a_i n^{-1} \log n} = a_i \sqrt{(a_i n)^{-1} \log n} \le a_i \sqrt{k^{-1} \log n} \le \sqrt{1/1200} \cdot a_i$ and $\sqrt{a_i n^{-1} \log n} \cdot \sqrt{a_i m} \le a_i \sqrt{(n/k) n^{-1} \log n} \le \sqrt{1/1200} \cdot a_i$. Therefore, $\theta \le (1 + 6\sqrt{2} \cdot 2\sqrt{1/1200}) a_i < \frac{3}{2} a_{\max}$, so $E[Y] \le l\theta \le \frac{3}{2} a_{\max} l$. Thus, by Corollary 2.3, we have that the probability that $Y > E[Y] + \sqrt{3 \cdot \frac{9}{2} \cdot \frac{3}{2} a_{\max} l \cdot \log n}$ is $O(1/n^{4.5})$. It follows that with probability $1 - O(m/n^{4.5}) = 1 - O(1/n^{3.5})$, for all i,

$$\begin{aligned}
I_i^1 &\le l a_i \sum_{j \ne i} a_j p_j + l \cdot 6\sqrt{2}\sqrt{a_i n^{-1} \log n}(1 + \sqrt{a_i m}) + \frac{9}{2}\sqrt{a_{\max} l \cdot \log n} \\
&\le l a_i \sum_{j \ne i} a_j p_j + 3\sqrt{2}\sqrt{a_i l \cdot \log n}(1 + \sqrt{a_i m}) + \frac{9}{2}\sqrt{a_{\max} l \cdot \log n} \\
&\le l a_i \sum_{j \ne i} a_j p_j + 9\sqrt{a_{\max} l \cdot \log n}(1 + \sqrt{a_{\max} m}) \\
&\le l a_i \sum_{j \ne i} a_j p_j + 18 a_{\max} \sqrt{ml \cdot \log n}.
\end{aligned}$$

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 with probability $1 - O(1/n^{3.5})$,

$$\begin{aligned}
|I_i^s - l a_i \sum_{j \ne i} a_j (p_{ij}^{>} + \frac{1}{2} p_{ij}^{=})| &\le 18 a_{\max} \sqrt{ml \log n} \qquad \text{for } s = 1, 3 \\
|I_i^s - l a_i \sum_{j \ne i} a_j (p_{ij}^{<} + \frac{1}{2} p_{ij}^{=})| &\le 18 a_{\max} \sqrt{ml \log n} \qquad \text{for } s = 2, 4.
\end{aligned}$$

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 in the initialization stage (see Section 1.2) to build the initial sets L_0, R_0. The second function, in Lemma 4.3, will be used to in the imbalance amplification stage to build the pair L_t, R_t from L_{t-1}, R_{t-1}.

Our cluster functions, defined on a set T, will depend on the edges between $v \in T$ and some set of vertices $T' \subseteq V - T$.

Lemma 4.2. Let u be a vertex from V_1, and let T be a random subset of $V - \{u\}$. Define $f(v) = d_{\{u\}}(v)$ for all $v \in T$. Then $c_1(f) = 2a_1(1-a_1)\Delta$ and $c_i(f) = -2a_1a_i\Delta$ for all $i > 1$.

Proof. Clearly,

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

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, $c_i(f) = -2a_1a_i\Delta$. ■

In the imbalance amplification stage (see Section 1.2), at each iteration we have a δ-unbalanced pair L', R', and the goal is to build a new δ-unbalanced pair L, R such that the imbalance of L, R is at least twice the imbalance of L', R'. To build this pair, we use procedure Partition on a set of vertices T and cluster function f, where T is a random subset of $V - (L' \cup R')$, and $f(v) = d_{L'}(v) - d_{R'}(v)$ for all $v \in T$. We now give some intuition on how the L, R-imbalances behave.

Denote $b_i = \mathrm{I}(V_i, L', R')$. For a pair of vertices $v \in V_i$ and $w \in V_j$, the probability that v wins in a duel with w depends on the value of $b_i - b_j$. More precisely, the winning probability of v behaves like the normal cumulative distribution function as a function of $b_i - b_j$. In particular, if $|b_i - b_j|$ is large enough, then the winning probability of v is close to 1 if $b_i - b_j > 0$, and close to 0 if $b_i - b_j < 0$. If $|b_i - b_j|$ is small enough, then the winning probability of v is roughly $\frac{1}{2} + \frac{1}{\sqrt{2\pi}}(b_i - b_j)$.

Now, suppose w.l.o.g. that $b_1 \geq b_2 \geq \cdots \geq b_m$. Since $b_1 - b_i$ is "large" for every $i > 2$ (recall that $b_i \leq b_2 \leq \delta b_1$ for $\delta < 1$), we have that the vertices of V_1 almost always win when they are in a duel against vertices of the other clusters. Thus, the L, R-imbalance of V_1 will be large, and V_1 will be the cluster with the largest L, R-imbalance. However, if $b_2 - b_3$ is also large, then the vertices of V_2 will almost always win when they are in a duel against vertices of the clusters $V_3, \ldots, V_m$. Thus, the L, R-imbalance of V_2 will also be large, and close to the L, R-imbalance of V_1. In other words, the pair L, R will not be δ-unbalanced. In order to avoid this problem, we will make sure that $b_2 - b_3$ is "small" by requiring an upper bound on $b_1 - b_m$. This upper bound causes all the winning probabilities to be approximately $\frac{1}{2} + \frac{1}{\sqrt{2\pi}}(b_i - b_j)$. The following lemma (combined with Theorem 4.1) will show that $\mathrm{I}(V_i, L, R) \approx ca_i \cdot b_i$ for some constant c that depends on p, r, m, and $|T|$. Thus, in the case of equal sized clusters, when moving from the pair L', R' to the pair L, R, the imbalances of the clusters are (approximately) multiplied by the same factor. Since L', R' is δ-unbalanced, we have that L, R is $(\delta + \epsilon)$-unbalanced for some small ϵ. Moreover, if $ca_1 \geq 2$ then the imbalance of L, R is at least twice the imbalance of L', R'.

We now formalize the ideas described above. Denote $\Gamma = \max_i |a_i - \frac{1}{m}|$.

Lemma 4.3. Let T' be a random set of vertices of size $2l$. Let L' and R' be two disjoint sets such that $T' = L' \cup R'$ and $|L'| = |R'|$. Let T be a random subset of $V - T'$ (of arbitrary size). Define $b_i = \mathrm{I}(V_i, L', R')$, and define $f(v) = d_{L'}(v) - d_{R'}(v)$ for all $v \in T$. Suppose that $p, r \in [\frac{1}{4}, \frac{3}{4}]$, $l \geq 7 \log n / a_{\max}$, and $\max_i b_i - \min_i b_i \leq \alpha / \Delta\sqrt{l}$ where $\alpha \leq 1$. Let

$$\beta = 2\Gamma + 5\sqrt{a_{\max} l^{-1} \log n}$$

and

$$\gamma = \sqrt{\frac{2}{\pi} / (\frac{1}{m} p(1-p) + (1 - \frac{1}{m}) r(1-r))}.$$

Then, with probability $1 - O(1/n^{3.5})$,

$$\left| c_i(f) - \gamma\Delta\sqrt{l} a_i \left(b_i - \sum_{j=1}^{m} a_j b_j \right) \right| \leq \gamma\Delta\sqrt{l} a_i (\beta\Delta + \frac{2}{9}\alpha^2)(\max_j b_j - \min_j b_j) + 3\frac{a_i}{\sqrt{l}}$$

for all i.

Proof. W.l.o.g., we assume that $b_1 \geq b_2 \geq \cdots \geq b_m$. By Corollary 2.4 (with $y = 2la_{\max}$), with probability $1 - O(1/n^{3.5})$, $|A'_j - 2la_j| \leq \sqrt{3 \cdot 4.5 \cdot 2a_{\max} l \log n}$ for all j (we use the fact that $l \geq 7 \log n / a_{\max}$). We will now assume that $|A'_j - 2la_j| \leq \sqrt{27 a_{\max} l \log n}$ for all j, and we will estimate $c_i(f)$ for some fixed i.

Recall that $c_i(f) = 2a_i \sum_{j \neq i} a_j (p^{>}_{ij} - p^{<}_{ij})$, so we need to estimate $p^{>}_{ij} - p^{<}_{ij}$ for all j. Fix some $j \neq i$, $v \in V_i \cap T$, and $w \in V_j \cap T$.

We denote $L' = \{u_1, \ldots, u_l\}$ and $R' = \{u_{l+1}, \ldots u_{2l}\}$. We have that $f(v) - f(w) = \sum_{s=1}^{4l} X_s$ where $X_1, \ldots, X_{4l}$ are independent random variables with $X_s = d_{\{u_s\}}(v)$ for $1 \leq s \leq l$, $X_s = -d_{\{u_s\}}(v)$ for $l+1 \leq s \leq 2l$, $X_s = -d_{\{u_{s-2l}\}}(w)$ for $2l+1 \leq s \leq 3l$, and $X_i = d_{\{u_{s-2l}\}}(w)$ for $3l+1 \leq s \leq 4l$. Clearly, $p^{>}_{ij} = P\left[\sum_{s=1}^{4l} X_s > 0\right]$. Recall that $\Phi(x)$ denotes the normal $(0,1)$ cumulative distribution function. Applying Theorem 2.1, we get that

$$\left| p^{>}_{ij} - \Phi\left(E\left[\sum_{s=1}^{4l} X_s\right] / \sqrt{B_{ij}} \right) \right| \leq \frac{4}{5} L_{ij}, \tag{1}$$

where $B_{ij} = \sum_{s=1}^{4l} \mathrm{Var}(X_s)$ and $L_{ij} = B_{ij}^{-3/2} \sum_{s=1}^{4l} E\left[|X_s - E[X_s]|^3\right]$. Similarly,

$$\left| p^{<}_{ij} - \left(1 - \Phi\left(E\left[\sum_{s=1}^{4l} X_s\right] / \sqrt{B_{ij}} \right) \right) \right| \leq \frac{4}{5} L_{ij}. \tag{2}$$

Combining (1) and (2) gives that

$$p^{>}_{ij} - p^{<}_{ij} \geq 2\Phi\left(E\left[\sum_{s=1}^{4l} X_s\right] / \sqrt{B_{ij}} \right) - 1 - \frac{8}{5} L_{ij}. \tag{3}$$

To finish the proof of the lemma, we will simplify (3): We will replace the terms $E\left[\sum_{s=1}^{4l} X_s\right]$, B_{ij}, and L_{ij} by other terms, and get rid of the Φ function.

We first compute the exact value of $E\left[\sum_{s=1}^{4l} X_s\right]$. For $s = 1, \ldots, m$, denote $A'_s = |V_s \cap T'|$. 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 variables $X_1, \ldots, X_l$, there are $\frac{A'_i + lb_i}{2}$ variables with $\mathrm{B}(p)$ distribution, and the rest of the variables have $\mathrm{B}(r)$ distribution. Out of $X_{l+1}, \ldots, X_{2l}$, $\frac{A'_i - lb_i}{2}$ 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,

$$\begin{aligned}
E\left[\sum_{s=1}^{4l} X_s\right] &= p\frac{A'_i + lb_i}{2} + r(l - \frac{A'_i + lb_i}{2}) - p\frac{A'_i - lb_i}{2} - r(l - \frac{A'_i - lb_i}{2}) \\
&\quad - p\frac{A'_j + lb_j}{2} - r(l - \frac{A'_j + lb_j}{2}) + p\frac{A'_j - lb_j}{2} + r(l - \frac{A'_j - lb_j}{2}) \\
&= plb_i - rlb_i - plb_j + rlb_j = \Delta l(b_i - b_j), \qquad (4)
\end{aligned}$$

where the second equality follows by canceling terms, and the last equality follows from the fact that $\Delta = p - r$.

Next, we give estimates on B_{ij} and L_{ij}. For a random variable X with distribution $\mathrm{B}(q)$ or $-\mathrm{B}(q)$, $\mathrm{Var}(X) = q(1-q)$. Thus, $B_{ij} = \sum_{s=1}^{4l} \mathrm{Var}(X_s) = (A'_i + A'_j)p(1-p) + (4l - A'_i - A'_j)r(1-r)$. 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

$$\frac{3}{4}l \leq B_{ij} \leq l. \qquad (5)$$

Similarly, define $h(x) = (1-x)^3 x + x^3(1-x)$, and we have that for a random variable X with distribution $\mathrm{B}(q)$ or $-\mathrm{B}(q)$, $E\left[|X_s - E[X_s]|^3\right] = h(q)$. Therefore,

$$L_{ij} = \frac{1}{B_{ij}^{3/2}} \sum_{s=1}^{4l} E\left[|X_s - E[X_s]|^3\right] = \frac{1}{B_{ij}^{3/2}} \left((A'_i + A'_j)h(p) + (4l - A'_i - A'_j)h(r)\right).$$

We have that $h(p) \leq \frac{1}{8}$ and $h(q) \leq \frac{1}{8}$, so

$$L_{ij} \leq \frac{1}{(\frac{3}{4}l)^{3/2}} \cdot 4l \cdot \frac{1}{8} = \frac{4}{3\sqrt{3}} \cdot \frac{1}{\sqrt{l}}. \qquad (6)$$

Combining Equations (3), (4), (5) and (6), we obtain that

$$p_{ij}^{>} - p_{ij}^{<} \geq 2\Phi\left(\Delta l(b_i - b_j)/\sqrt{B_{ij}}\right) - 1 - \frac{3}{2}/\sqrt{l}. \qquad (7)$$

We now wish to get rid of the Φ in Equation (7). We do this by using the fact that $\Phi(x) \approx \frac{1}{2} + \frac{1}{\sqrt{2\pi}}x$ near 0. More precisely, 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 $0 \leq \frac{\Delta l(b_i - b_j)}{\sqrt{B_{ij}}} \leq \frac{\Delta l(b_i - b_j)}{\sqrt{3l/4}} \leq \frac{2}{\sqrt{3}}\alpha$, then for $j > i$,

$$p_{ij}^{>} - p_{ij}^{<} \geq \sqrt{\frac{2}{\pi}} \frac{\Delta l(b_i - b_j)}{\sqrt{B_{ij}}}(1 - \frac{2}{9}\alpha^2) - \frac{3}{2\sqrt{l}}. \qquad (8)$$

Moreover, for $x \in [0,-1]$, $\Phi(x) \geq \frac{1}{2} + \frac{1}{\sqrt{2\pi}}x$, so for $i < j$,

$$p_{ij}^{>} - p_{ij}^{<} \geq \sqrt{\frac{2}{\pi}} \frac{\Delta l(b_i - b_j)}{\sqrt{B_{ij}}} - \frac{3}{2\sqrt{l}}. \tag{9}$$

Finally, in order to combine the lower bounds on $p_{ij}^{>} - p_{ij}^{<}$ into a lower bound on $c_i(f)$, we want to replace the term B_{ij} in (8) and (9) by a common term B that does not depend on i or j. In order to perform this replacement, we define $B = 4l\frac{1}{m}p(1-p) + 4l(1-\frac{1}{m})r(1-r)$, and need to show that B is close to B_{ij} for all j (more precisely, we need to bound $|1/\sqrt{B_{ij}} - 1/\sqrt{B}|$).

For every s, $|A'_s - 2la_s| \leq \sqrt{27a_{\max} l \log n}$ so $|A'_s - \frac{2l}{m}| \leq |A'_s - 2la_s| + 2l|a_s - \frac{1}{m}| \leq \sqrt{27a_{\max} l \log n} + 2\Gamma l < \frac{9}{8}\beta l$. Therefore,

$$|B_{ij} - B| = \left| \left(A'_i + A'_j - \frac{4l}{m} \right) (p(1-p) - r(1-r)) \right| \leq 2 \cdot \frac{9}{8}\beta l \cdot \Delta |1-p-r| \leq \frac{9}{8}\beta \Delta l,$$

where the last inequality follows from the fact that $p, r \in [\frac{1}{4}, \frac{3}{4}]$. Since $B, B_{ij} \geq \frac{3}{4}l$, we conclude that $|B_{ij} - B| \leq \frac{9}{8}\beta\Delta \cdot \frac{4}{3}B_{ij} = \frac{3}{2}\beta\Delta B_{ij}$ and $|B_{ij} - B| \leq \frac{3}{2}\beta\Delta B$. Thus, $\frac{B}{B_{ij}} \leq 1 + \frac{3}{2}\beta\Delta$ and $\frac{B_{ij}}{B} \leq 1 + \frac{3}{2}\beta\Delta$. Using the fact that $\sqrt{1+x} \leq 1 + \frac{1}{2}x$ for all x, we obtain that if $B \geq B_{ij}$ then

$$\left| \frac{1}{\sqrt{B_{ij}}} - \frac{1}{\sqrt{B}} \right| = \frac{1}{\sqrt{B}} \left(\sqrt{\frac{B}{B_{ij}}} - 1 \right) \leq \frac{1}{\sqrt{B}} \left(\sqrt{1 + \frac{3}{2}\beta\Delta} - 1 \right) \leq \frac{3\beta\Delta}{4\sqrt{B}} < \frac{\beta\Delta}{\sqrt{B}},$$

and if $B < B_{ij}$ then

$$\left| \frac{1}{\sqrt{B_{ij}}} - \frac{1}{\sqrt{B}} \right| \leq \frac{1}{\sqrt{B_{ij}}} \left(\sqrt{1 + \frac{3}{2}\beta\Delta} - 1 \right) \leq \frac{\beta\Delta}{\sqrt{B_{ij}}} \leq \frac{\beta\Delta}{\sqrt{B}}.$$

Combining the two inequalities above with Equations (8) and (9), we obtain that for $j > i$,

$$\begin{aligned} p_{ij}^{>} - p_{ij}^{<} &\geq \sqrt{\frac{2}{\pi}} \frac{\Delta l(b_i - b_j)}{\sqrt{B}} (1 - \beta\Delta)(1 - \frac{2}{9}\alpha^2) - \frac{3}{2\sqrt{l}} \\ &\geq \sqrt{\frac{2}{\pi}} \frac{\Delta l(b_i - b_j)}{\sqrt{B}} (1 - \beta\Delta - \frac{2}{9}\alpha^2) - \frac{3}{2\sqrt{l}} \\ &= \frac{\gamma}{2}\Delta\sqrt{l}(b_i - b_j)(1 - \beta\Delta - \frac{2}{9}\alpha^2) - \frac{3}{2\sqrt{l}}, \end{aligned}$$

and for $i < j$,

$$p_{ij}^{>} - p_{ij}^{<} \geq \frac{\gamma}{2}\Delta\sqrt{l}(b_i - b_j)(1 + \beta\Delta) - \frac{3}{2\sqrt{l}}.$$

We now use the lower bounds on $p_{ij}^{>} - p_{ij}^{<}$ to obtain a lower bound on $c_i(f)$:

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

Using similar arguments we obtain a matching upper bound on $c_i(f)$. ∎

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 $\Gamma = O(1/m\log n)$. We use algorithm Solve$_3$ with a procedure Find that is described below.

Recall that the goal of procedure Find is to find a subcluster of G. We first show that given a $\frac{1}{10}$-unbalanced pair L, R whose imbalance is large enough, we can generate a subcluster of G using the following procedure (the input to the procedure is a cluster graph G, and disjoint sets of vertices L, R, and S):

```
Subcluster($G, L, R, S$):
1: 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 $f(v) = d_L(v) - d_R(v)$.
2: Let $j$ be the first index for which $f(v_j) - f(v_{j+1}) > 15\sqrt{|L|\cdot\log n}$.
3: **Return** $\{v_1, \ldots, v_j\}$.
```

Lemma 5.1. *Let L and R be disjoint sets of vertices with $|L| = |R|$, and let S be a random subset of $V-(L\cup R)$ of size at least $7m\log n$. Let $b_i = \mathrm{I}(V_i, L, R)$ and w.l.o.g. $b_1 \geq b_2 \geq \cdots \geq b_m$. If $b_1 - b_2 > 30\sqrt{\log n}/\Delta\sqrt{|L|}$ then procedure Subcluster returns the subcluster $V_1 \cap S$ with probability $1 - O(1/n^3)$.*

Proof. Define $l = |L|$ and $D = 45\sqrt{l\cdot\log n}$. The failure events of procedure Subcluster are:

1. $S \cap V_i = \emptyset$ for some i.

2. $|f(v) - E[f(v)]| > \frac{1}{6}D$ for some vertex $v \in S$.

The probability that event 1 occurs is $o(n^{-3})$. The probability that event 2 happens is $o(n^{-3})$ (this follows from writing $f(v)$ as a sum of 0/1 random variables minus a sum of 0/1 random variables, and applying Corollary 2.3 on each of these summations). For the rest of the proof assume that none of these events happens.

For $v \in V_i$, $E[f(v)] = \Delta l b_i$ (see the proof of Lemma 4.3). Since $\Delta l b_1 - \Delta l b_2 > \Delta l \cdot (30\sqrt{\log n}/\Delta\sqrt{l}) = \frac{2}{3}D$, it follows that $\min_{v \in V_1 \cap S} f(v) - \max_{v \in S - V_1} f(v) \geq (\Delta l b_1 - \frac{1}{6}D) - (\Delta l b_2 + \frac{1}{6}D) > \frac{1}{3}D$. Furthermore, for $v, v' \in V_1 \cap S$, $|f(v) - f(v')| \leq \frac{1}{3}D$. Therefore, $\{v_1, \ldots, v_j\} = V_1 \cap S$. ■

Procedure Find will build a $\frac{1}{10}$-unbalanced pair with imbalance at least $\frac{10}{9} \cdot 30\sqrt{\log n}/\Delta\sqrt{|L|}$ and then use procedure Subcluster to find a subcluster. As outlined in Section 1.2, building this pair consists of two stages. In the initialization stage, procedure Find builds two initial sets L_0 and R_0 using procedure Partition and the cluster function of Lemma 4.2. In the imbalance amplification stage, a series of pairs L_t, R_t is generated, where L_t, R_t is generated from L_{t-1}, R_{t-1} using procedure Partition and the cluster function of Lemma 4.3. Each pair L_t, R_t is $\frac{1}{10}$-unbalanced, and moreover, the imbalance L_t, R_t is at least twice the imbalance of L_{t-1}, R_{t-1}. If we are able to build such a series, then at some point we will have a $\frac{1}{10}$-unbalanced pair L_t, R_t with imbalance at least $\frac{10}{9} \cdot 30\sqrt{|L| \cdot \log n}$, and then we will use procedure Subcluster to find a subcluster (Lemma 5.1).

The algorithm outlined above has a crucial problem: In order to use Lemma 4.3, we need the imbalance of the pair L_t, R_t to be at most $c/\Delta\sqrt{|L_t|}$ for some constant $c \leq 1$. The imbalance amplification stage might generate a pair L_t, R_t whose imbalance is more than $1/\Delta\sqrt{|L_t|}$ but less than $\frac{10}{9} \cdot 30\sqrt{\log n}/\Delta\sqrt{|L_t|}$. In this situation, we can neither generate a new pair L_{t+1}, R_{t+1} or obtain a subcluster from the current pair. The solution to this problem is to build at each iteration an additional $\frac{1}{10}$-unbalanced pair $\hat{L}_t, \hat{R}_t$. The pair $\hat{L}_t, \hat{R}_t$ is built using procedure Partition and the same cluster function that is used to build L_t, R_t, and therefore the imbalance of $\hat{L}_t, \hat{R}_t$ is approximately the same as the imbalance of L_t, R_t. By making the sets $\hat{L}_t$ and $\hat{R}_t$ bigger than L_t and R_t by a factor of $\Omega(\log n)$, we obtain that at every step, either

- The imbalance of L_t, R_t is at most $c/\Delta\sqrt{|L_t|}$, and we can be build a new pair L_{t+1}, R_{t+1}, or
- The imbalance of L_t, R_t is greater than $c/\Delta\sqrt{|L_t|}$. In this case, the imbalance of $\hat{L}_t, \hat{R}_t$ is $\Omega(1/\Delta\sqrt{|L_t|}) = \Omega(\sqrt{\log n}/\Delta\sqrt{|\hat{L}_t|})$ and procedure Subcluster, applied on $\hat{L}_t, \hat{R}_t$, can build a subcluster.

Formally, procedure Find is as follows:

```
   Find$(G, V')$:
1: Let $\hat{l} = 10^{15}(m^2\Delta^{-2}\log^2 n + m\log^5 n)$, $l = \hat{l}/(5 \cdot 10^6 \log n)$, $l' = \hat{l}/(5 \cdot 10^6 \log^2 n)$,
   $s = 2500m\Delta^{-2}\log n$, and $D = 45\sqrt{\hat{l}\log n}$. Let $l_t = l$ for $0 \leq t \leq 2$ and $l_t = l'$ for
   $t \geq 3$.
2: Randomly select disjoint sets $\hat{W}$ and $W_0$ of vertices from $V'$ of sizes $2\hat{l}$ 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**
```

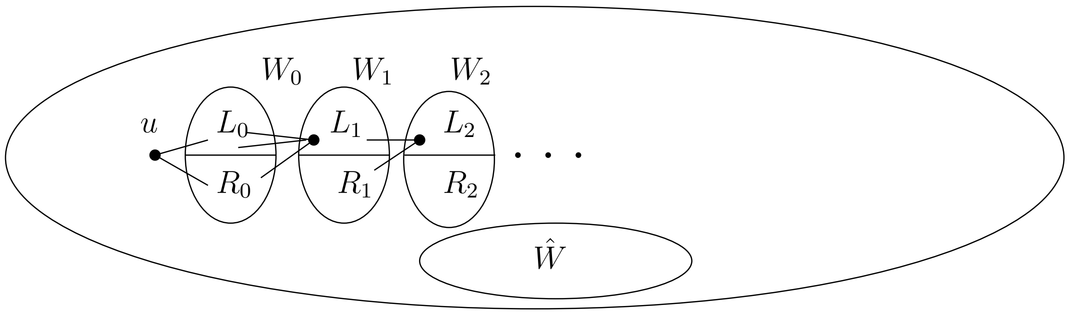

Figure 2: The sets created by procedure Find.

```
6:     Randomly select disjoint sets $W_{t+1}$ and $S_t$ of unchosen vertices from $V'$ of
       sizes $2l_{t+1}$ and $s$, respectively.
7:     Let $v_1, \ldots, v_s$ be an ordering of $S_t$ such that $f_t(v_1) \geq f_t(v_2) \geq \cdots \geq f_t(v_s)$,
       where $f_t(v) = d_{\hat{L}_t}(v) - d_{\hat{R}_t}(v)$.
8:     **If** $\max_j\{f_t(v_j) - f_t(v_{j+1})\} > D$
9:         **Return** Subcluster$(G, \hat{L}_t, \hat{R}_t, S_t)$.
10:    $\hat{L}_{t+1}, \hat{R}_{t+1} \leftarrow$ Partition$(G, \hat{W}, d_{L_t} - d_{R_t})$.
11:    $L_{t+1}, R_{t+1} \leftarrow$ Partition$(G, W_{t+1}, d_{L_t} - d_{R_t})$.
12: **Output** 'Failure'.
```

See also Figure 2. The total number of vertices selected by procedure Find is at most $2\hat{l} + 1 + s \log n + 2\sum_{t=0}^{\log n+1} l_t \leq 3\hat{l}$, and this number can be more than $n/3$ for some choice of parameters, making the procedure inapplicable. However, we assume that $k \geq 10^8 \Delta^{-1}\sqrt{n}\log n$, and in that case, $m \leq \frac{n}{k} \leq \Delta\sqrt{n}/(10^8 \log n)$. Therefore, $3\hat{l} \leq n/3$.

Note that in the first three iterations of Steps 6–11, 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).

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 $\hat{r}_i^t = \hat{b}_i^t/\hat{b}_1^t$. We also denote $c_i^0 = c_i(d_{\{u\}})$ and $c_i^t = c_i(d_{L_t} - d_{R_t})$ for $t \geq 1$. 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 $t^* \geq 4$ and $m \geq 3$. The cases $t^* \leq 3$ or $m = 2$ are simpler and we omit the analysis for these cases.

In the next lemmas we analyze procedure Find. To prove its correctness, we need to show that (1) Procedure Find executes step 9, and (2) When procedure Find executes step 9, the condition of Lemma 5.1 is satisfied. In order to show (1), we will show that $b_1^t \geq e \cdot b_1^{t-1}$, for all t. Since $\frac{1}{n} \leq b_1^t \leq 1$, it follows that procedure Find must stop after at most $\log n$ iterations. In order to show (2), we will show that all the pairs L_t, R_t are $\frac{1}{10}$-unbalanced, that is $r_i^t \leq \frac{1}{10}$ for all t and $i > 1$. More specifically, we will show that $r_i^0 \leq \frac{1}{100}$ for all $i > 1$, and moreover, $|r_i^t - r_i^0| \leq \frac{9}{100}$ for all t and $i > 1$. Note that Property 1 is trivially satisfied as there are no important edges (the cluster V_1 is determined only by the random choice of u).

Define $d_t = \frac{1}{100} \cdot \frac{\Delta}{m}$ for $t \leq 2$, and $d_t = \frac{1}{100} \cdot \frac{\Delta}{m}\sqrt{\log n}$ for $t \geq 3$. The following events can cause procedure Find to fail:

1. $|b_i^t - c_i^t| > d_t$ for some i and t.
2. $|\hat{b}_i^t - c_i^t| > d_t$ for some i and t.
3. $|V_i \cap S_t| < \frac{1}{2}a_i s$ for some i and t.
4. $|f_t(v) - E[f_t(v)]| \leq \frac{1}{6}D$ for some t and $v \in S_t$.

The following Lemma bounds the probability of these events.

Lemma 5.2. Suppose that $\Gamma \leq 1/(10000 m \log n)$, $k \geq 10^8 \Delta^{-1}\sqrt{n}\log n$, and $p, r \in [\frac{1}{4}, \frac{3}{4}]$. Then, the probability that at least one of the events 1–4 happens is $O(n^{-3})$.

Proof. For fixed t, we have from Theorem 4.1 that the probability that $|b_i^t - c_i^t| > 72 a_{\max}\sqrt{\frac{m}{l}\log n}$ for some i is $O(n^{-3.5})$. Since $72 a_{\max}\sqrt{\frac{m}{l}\log n} = (1+o(1))72\sqrt{\frac{1}{ml}\log n} \leq \frac{1}{100}\cdot\frac{\Delta}{m}$, it follows that event 1 happens with probability $O(\log n \cdot n^{-3.5}) = o(n^{-3})$. The same arguments gives that event 2 happens with probability $o(n^{-3})$.

By Corollary 2.4, the probability that event 3 happens is $o(n^{-3})$, and by Corollary 2.3, the probability that event 4 happens is $o(n^{-3})$. ∎

For the rest of this section, we assume that events 1–4 do not happen. We first analyze step 4 of procedure Find.

Lemma 5.3. Under the conditions of Lemma 5.2, with probability $1 - O(1/n^3)$,

1. $b_1^0, \hat{b}_1^0 \geq (1 - \frac{1}{100})\frac{4}{3}\frac{\Delta}{m}$.
2. For all $i > 1$, $-\frac{1}{2} - \frac{1}{50} \leq r_i^0, \hat{r}_i^0 \leq \frac{1}{100}$.

Proof. In the proof of lemma (and also in the proofs of the next lemmas), we only prove the bounds on b_i^0 and r_i^0, as the bounds on $\hat{b}_i^0$ and $\hat{r}_i^0$ are similar.

(1) From Lemma 4.2, we have that

$$c_1^0 = 2a_1(1-a_1)\Delta \geq 2(1-o(1))\frac{1}{m}(1-(1+o(1))\frac{1}{m})\Delta \geq (1-o(1))\frac{4}{3}\frac{\Delta}{m}.$$

Therefore, $b_1^0 \geq c_1^0 - d_1^0 \geq (1 - \frac{1}{100})\frac{4}{3}\frac{\Delta}{m}$.

(2) By Lemma 4.2, $c_i^0 = -2a_1 a_i \Delta$ for every $i > 1$. Therefore, $b_i^0 \leq c_i^0 + d_0 \leq d_0 \leq \frac{1}{100}\cdot\frac{\Delta}{m}$. Thus, $r_i^0 \leq \frac{1}{100}$. 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 $b_i^0 \geq c_i^0 - d_0 \geq -(1+\frac{1}{50})\frac{2}{3}\frac{\Delta}{m}$. Thus, $r_i^0 \geq -(1+\frac{1}{50})\frac{2}{3}\frac{\Delta}{m}/((1-\frac{1}{100})\frac{4}{3}\frac{\Delta}{m}) \geq -\frac{1}{2} - \frac{1}{50}$. ∎

Next, we analyze the first three iterations of steps 6–11.

Lemma 5.4. Under the conditions of Lemma 5.2, with probability $1 - O(1/n^3)$,

1. For all $0 \leq t \leq 3$, $b_1^t \geq \log^{t/2} n \cdot b_1^0$ and $\hat{b}_1^t \geq \log^{t/2} n \cdot \hat{b}_1^0$.
2. For all $0 \leq t \leq 3$ and all $i > 1$, $|r_i^t - r_i^0| \leq \frac{t}{100}$ and $|\hat{r}_i^t - \hat{r}_i^0| \leq \frac{t}{100}$.

Note that by combining Lemma 5.3(2) and Lemma 5.4(2) we obtain that $-\frac{1}{2} - \frac{1}{10} \leq r_i^t \leq \frac{1}{10}$.

Proof. The lemma is proved using induction on t. The base $t = 0$ it true due to Lemma 5.3. Assuming we proved (1) and (2) for t, we will prove these bounds for $t + 1$. For the rest of the proof, we assume w.l.o.g. that $b_1^t \geq b_2^t \geq \cdots \geq b_m^t$.

(1) From the induction hypothesis, it suffices to prove that $b_1^{t+1} \geq \sqrt{\log n} \cdot b_1^t$. For a vertex $v \in V_i \cap S_t$, $E[f_t(v)] = \Delta\hat{l}\hat{b}_i^t$. 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 $f_t(v)$ for $v \in V_1 \cap S_t$ and the maximum value of $f_t(v')$ for $v' \in S_t - V_1$ would be at least $2D - 2 \cdot \frac{1}{6}D > D$, contradicting the fact that the algorithm did not stop at iteration t. From the fact that $|b_i^t - b_j^t| \leq 2d_t$, $|\hat{b}_i^t - \hat{b}_j^t| \leq 2d_t$, and $r_i^t \geq -\frac{1}{2} - \frac{1}{10}$ (the latter is true due to the induction hypothesis) we have that for all i and j, $|b_i^t - b_j^t| \leq |\hat{b}_i^t - \hat{b}_j^t| + \frac{4}{100} \cdot \frac{\Delta}{m} \leq \frac{16}{10}\hat{b}_1^t + \frac{2}{50} \cdot \frac{\Delta}{m}$. By the induction hypothesis $\hat{b}_1^t \geq \frac{\Delta}{m}$ and $\hat{b}_1^t \leq \frac{10}{9}(\hat{b}_1^t - \hat{b}_2^t)$. Therefore, for all i and j, $|b_i^t - b_j^t| \leq (\frac{16}{10} + \frac{2}{50})\hat{b}_1^t < 2(\hat{b}_1^t - \hat{b}_2^t) \leq 4D/\Delta\hat{l} = 180\sqrt{\log n}/\Delta\sqrt{\hat{l}}$. As $\hat{l} = 5 \cdot 10^6 \cdot l \log n$, we conclude that for all i and j, $|b_i^t - b_j^t| \leq \frac{180}{\sqrt{5 \cdot 10^6}} \cdot \frac{1}{\Delta\sqrt{l}} < \frac{1}{12} \cdot \frac{1}{\Delta\sqrt{l}}$.

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

$$b_1^{t+1} \geq F_0 a_1 \left(b_1^t - \sum_{j=1}^{m} a_j b_j^t - \left(2\Gamma\Delta + 5\sqrt{\frac{2}{ml}\log n} \cdot \Delta + \frac{2}{9} \cdot \frac{1}{12^2} \right) (b_1^t - b_m^t) \right) - 3\frac{a_1}{\sqrt{l}} - d_1^{t+1}.$$

We now give bounds for the terms in the inequality above.

- $(1 - o(1))F \leq F_0 a_1 \leq (1 + o(1))F$ (follows from the fact that $\Gamma = o(1)$).
- Since $r_i^t \geq -\frac{1}{2} - \frac{1}{10}$, it follows that $|b_j^t| \leq b_1^t$ for all j. Using the fact that $\sum_{j=1}^m b_j^t = 0$, we obtain that
$$\sum_{j=1}^{m} a_j b_j^t = \sum_{j=1}^{m} \left(a_j - \frac{1}{m} \right) b_j^t \leq \sum_{j=1}^{m} \left| a_j - \frac{1}{m} \right| b_1^t \leq m\Gamma b_1^t = o(b_1^t).$$
- $2\Gamma\Delta = o(1)$ (follows from the fact that $\Gamma = o(1)$).
- $5\sqrt{\frac{2}{ml}\log n} \cdot \Delta = o(1)$ (follows from the fact that $l \geq 2 \cdot 10^8 \log^4 n$).
- $b_1 - b_m \leq 2b_1$ (follows from the fact that $r_i^t \geq -\frac{1}{2} - \frac{1}{10}$).
- From the fact that $F = \gamma\Delta\sqrt{l}/m \geq \frac{3}{2}\Delta\sqrt{l}/m > 10000\sqrt{\log n}$ and $b_1^t \geq \log^{t/2} n \cdot \Delta/m$ we conclude that
$$3\frac{a_1}{\sqrt{l}} < \frac{3}{\sqrt{l}} = O\left(\frac{1}{\sqrt{\log n}} \frac{\Delta}{m} \right) = o(Fb_1^t),$$
and
$$d_1^{t+1} = o(Fb_1^t).$$

Combining the bounds above we get

$$b_1^{t+1} \geq (1-o(1))Fb_1^t - (1+o(1))\frac{1}{324}Fb_1^t - o(Fb_1^t) > \left(1-\frac{1}{200}\right)Fb_1^t > \sqrt{\log n}\cdot b_1^t.$$

(2) Using the induction hypothesis, we need to show that $|r_i^{t+1}-r_i^t| \leq \frac{1}{100}$. In the first part we have shown that $b_1^{t+1} \geq (1-\frac{1}{200})Fb_1^t$. Using similar arguments we obtain that $b_1^{t+1} \leq (1+\frac{1}{200})Fb_1^t$, or in other words, $|b_1^{t+1}-Fb_1^t| \leq \frac{1}{200}Fb_1^t$. Similarly, $|b_i^{t+1}-Fb_i^t| \leq \frac{1}{200}Fb_1^t$. Therefore,

$$\begin{aligned}\left|\frac{b_i^{t+1}}{b_1^{t+1}} - r_i^t\right| &\leq \frac{\left|b_i^{t+1}-Fb_i^t\right| + \left|Fb_i^t - r_i^t b_1^{t+1}\right|}{b_1^{t+1}} = \frac{\left|b_i^{t+1}-Fb_i^t\right| + r_i^t\left|Fb_1^t - b_1^{t+1}\right|}{b_1^{t+1}} \\ &\leq \frac{(1+\frac{1}{10})\frac{1}{200}Fb_1^t}{(1-\frac{1}{200})Fb_1^t} < \frac{1}{100}.\end{aligned}$$ ■

Finally, we analyze the remaining iterations of steps 6–11.

Lemma 5.5. Under the conditions of Lemma 5.2, with probability $1-O(1/n^3)$,

1. For $4 \leq t \leq t^*$, $b_1^t \geq 2^{t-3}\log^{3/2} n \cdot b_1^0$ and $\hat{b}_1^t \geq 2^{t-3}\log^{3/2} n\cdot \hat{b}_1^0$.
2. For all $4 \leq t \leq t^*$ and all $i > 1$, $|r_i^t - r_i^0| \leq \frac{t-3}{100\log n} + \frac{3}{100}$ and $|\hat{r}_i^t - \hat{r}_i^0| \leq \frac{t-3}{100\log n} + \frac{3}{100}$.

Note that by combining Lemma 5.3(2), and Lemma 5.5(2) we obtain that $-\frac{1}{2}-\frac{1}{10} \leq r_i^t \leq \frac{1}{10}$.

Proof. Again we prove the lemma using induction. Assume we proved (1) and (2) for t. W.l.o.g. $b_1^t \geq b_2^t \geq \cdots \geq b_m^t$.

(1) It suffices to prove that $b_1^{t+1} \geq 2b_1^t$. Similarly to the proof of Lemma 5.4, we have that for all i and j, $|b_i^t - b_j^t| \leq 180\sqrt{\log n}/\Delta\sqrt{\hat{l}}$, and since $\hat{l} = 5\cdot 10^6 \cdot l'\log^2 n$, we conclude that $|b_i^t - b_j^t| \leq \frac{180}{\sqrt{5\cdot 10^6\log n}}\cdot\frac{1}{\Delta\sqrt{l'}} \leq \frac{1}{12\sqrt{\log n}}\cdot\frac{1}{\Delta\sqrt{l'}}$.

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

$$\begin{aligned} b_1^{t+1} \geq F_0' a_1 &\left(b_1^t - \sum_{j=1}^m a_j b_j^t - \left(2\Gamma\Delta + 5\sqrt{\frac{2}{ml'}\log n}\cdot\Delta + \frac{2}{9}\cdot\frac{1}{12^2\log n}\right)(b_1^t - b_m^t)\right) \\ &- 3\frac{a_1}{\sqrt{l'}} - d_1^{t+1}.\end{aligned}$$

Now,

- $\left(1-\frac{1}{10^4\log n}\right)F' \leq F_0' a_1 \leq \left(1+\frac{1}{10^4\log n}\right)F'$.
- $\sum_{j=1}^m a_j b_j^t \leq m\Gamma b_1^t \leq \frac{1}{10^4\log n} b_1^t$.
- $2\Gamma\Delta \leq \frac{2}{10^4\log n}$.
- $5\sqrt{\frac{2}{ml'}\log n}\cdot\Delta \leq \frac{5}{10^4\log n}$.

- $b_1^t - b_m^t \leq 2b_1^t$.
- $3\frac{a_1}{\sqrt{l'}} = O\left(\frac{1}{\sqrt{l'}}\right) = O\left(\frac{\Delta}{m}\right) = o\left(\frac{1}{\log n}F'b_1^t\right)$, where the last equality follows from the fact that $F' = \gamma\Delta\sqrt{l'}/m \geq \frac{3}{2}\Delta\sqrt{l'}/m > 10000$ and $b_1^t \geq \log^{3/2} n \cdot \Delta/m$.
- $d_1^{t+1} = \frac{1}{100} \cdot \frac{\Delta}{m}\sqrt{\log n} \leq \frac{1}{10^6 \cdot \log n} \cdot F'b_1^t$.

Thus,

$$\begin{aligned} b_1^{t+1} &\geq \left(1 - \frac{1}{10^4 \log n}\right) F'b_1^t - (1+o(1))\left(\frac{1+4+10}{10^4} + \frac{1}{324} + \frac{1}{10^6}\right)\frac{1}{\log n}F'b_1^t - o(F'b_1^t) \\ &\geq \left(1 - \frac{1}{200\log n}\right) F'b_1^t. \end{aligned}$$

In particular, $b_1^{t+1} \geq 2b_1^t$.

(2) Similarly to the proof of Lemma 5.4, $|b_1^{t+1} - Fb_1^t| \leq \frac{1}{200\log n}Fb_1^t$. Similarly, $|b_i^{t+1} - Fb_i^t| \leq \frac{1}{200\log n}Fb_1^t$. Therefore, $|r_i^{t+1} - r_i^t| \leq \frac{1}{100\log n}$. ■

Lemma 5.6. Under the conditions of Lemma 5.3, with probability $1 - O(1/n^3)$, procedure Find returns a subcluster of V_1 of size at least $1200\log n/\Delta^2$. The running time of procedure Find is $O(m^3\Delta^{-4}\log^3 n \cdot (m + \log n) + m^2 \log^9 n)$.

Proof. By Lemma 5.5, for all $4 \leq t \leq t^*$, $b_1^t > 2^t \cdot b_1^0$. We have that $b_1^0 > \frac{1}{n}$, so $b_1^t > 2^t/n$. Since $b_1^t \leq 1$ by definition, we conclude that $t^* < \log n$, namely procedure Find stops at Step 9. We now look at the last iteration of procedure Find. W.l.o.g. assume that $\hat{b}_1^{t^*} \geq \hat{b}_2^{t^*} \geq \cdots \geq \hat{b}_m^{t^*}$.

For $v \in V_i$, $E\left[f_{t^*}(v)\right] = \Delta\hat{l}\hat{b}_i^{t^*}$. As $\max_j\{f_{t^*}(v_j) - f_{t^*}(v_{j+1})\} > D$, we have that there is an index i such that $\Delta\hat{l}\hat{b}_i^{t^*} - \Delta\hat{l}\hat{b}_{i+1}^{t^*} > \frac{2}{3}D$. By Lemma 5.5, $\Delta\hat{l}(\hat{b}_1^{t^*} - \hat{b}_2^{t^*}) \geq \Delta\hat{l}(\hat{b}_i^{t^*} - \hat{b}_{i+1}^{t^*}) > \frac{2}{3}D$. Therefore, procedure Subcluster returns the subcluster $V_1 \cap S_{t^*}$ (Lemma 5.1). Finally, we have that $|V_1 \cap S_{t^*}| \geq \frac{1}{2}a_1 s \geq 1200\log n/\Delta^2$.

We now compute the time complexity of Find: In a single iteration, the time to compute $f_t(v)$ for all $v \in S_t$ 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

$$O(s\hat{l}\log n + l\hat{l} + l'\hat{l}\log n) = O\left(\frac{m^3}{\Delta^4}\log^3 n \cdot (m + \log n) + m^2\log^9 n\right). \qquad ■$$

Lemma 5.7. $\Gamma \leq (\max_i |V_i|/k - 1)/m$.

Proof. Denote $r = \max_i |V_i|/k = a_{\max}/a_{\min}$. By definition, $\Gamma = \max\{a_{\max} - \frac{1}{m}, \frac{1}{m} - a_{\min}\}$. Clearly,

$$a_{\max} = ra_{\min} \leq \frac{r}{m}$$

and

$$a_{\min} = \frac{1}{r}a_{\max} \geq \frac{1}{r} \cdot \frac{1}{m} \geq (1 - (r-1))\frac{1}{m}.$$

Therefore, $a_{\max} - \frac{1}{m} \leq (r-1)/m$ and $\frac{1}{m} - a_{\min} \leq (r-1)/m$. ■

We are now ready to prove Theorem 1.1.

Proof (Theorem 1.1). Let G be a cluster graph that satisfies the requirements of Theorem 1.1 and requirements (R1)–(R3) from Section 3. From Lemma 5.7 we get that for every cluster collection G' of G,

$$\begin{aligned}\Gamma_{G'} &\le \max_i |V_{i,G'}|/k_{G'} \le \max_i |V_i|/k \\ &\le 1 + 1/(10000 m \log n) \le 1 + 1/(10000 m_{G'} \log n_{G'}).\end{aligned}$$

It follows that every cluster collection of G satisfies the requirements of Lemma 5.6. Theorem 1.1 now follows from Lemma 5.6 and Lemma 3.3. ■

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 U_1 and U_2, and then selects u to be the vertex from U_1 with highest d_{U_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$_2(G, V', \epsilon)$:
1:  Let $\lambda = 2^{3^{\lceil 1/\epsilon \rceil + 1}}$, $\hat{l} = 10^{10}\lambda^{15}(m\Delta^{-2}\log^2 n + m^2\Delta^{-2(1+\epsilon)}\log n)$, $l = \hat{l}/(10^6\lambda^{11}\log n)$,
    $s = 2500m\Delta^{-2}\log n$, $D = 50\lambda^2\sqrt{\hat{l}\log n}$, $u_1 = 3m\log n$ and $u_2 = \frac{1}{100}n$.
2:  Randomly select disjoint sets $U_1$, $U_2$, $\hat{W}$, and $W_0$ of vertices from $V'$ of sizes $s$,
    $u_1$, $u_2$, $2\hat{l}$, and $2l$, respectively.
3:  Let $u$ be a vertex in $U_1$ such that $d_{U_2}(u)$ is maximum.
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** $t = 0, 1, \ldots, \lceil 1/\epsilon \rceil$ **do**
6:      Randomly select disjoint sets $W_{t+1}$ and $S_t$ of unchosen vertices from $V'$ of
        sizes $2l$ and $s$, respectively.
7:      Let $v_1, \ldots, v_s$ be an ordering of $S_t$ such that $f_t(v_1) \ge f_t(v_2) \ge \cdots \ge f_t(v_s)$,
        where $f_t(v) = d_{\hat{L}_t}(v) - d_{\hat{R}_t}(v)$.
8:      **If** $\max_j\{f_t(v_j) - f_t(v_{j+1})\} > D$
9:          Let $j$ be the first index for which $f_t(v_j) - f_t(v_{j+1}) > \frac{2}{5\lambda^2}D$,
10:         **Return** $\{v_1, \ldots, v_j\}$.
11:     $\hat{L}_{t+1}, \hat{R}_{t+1} \leftarrow$ Partition$(G, \hat{W}, d_{L_t} - d_{R_t})$.
12:     $L_{t+1}, R_{t+1} \leftarrow$ 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 $u_1 +$

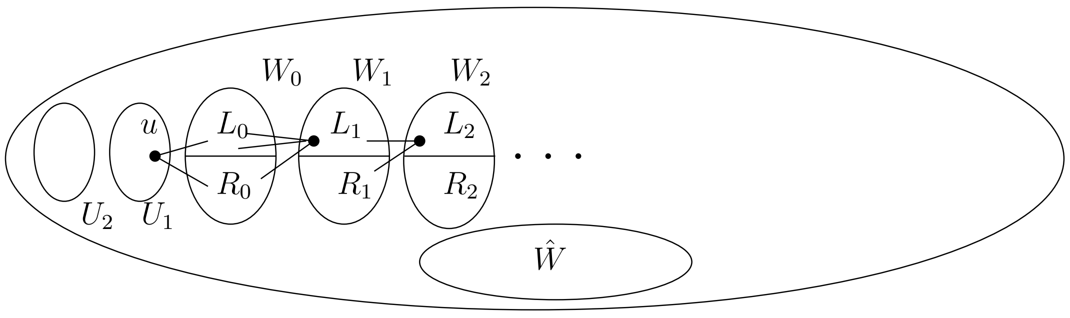

Figure 3: The sets created by procedure Find$_2$.

$u_2 + 2\hat{l} + (\lceil 1/\epsilon \rceil + 1) \cdot (2l + s)$. We assume that the bound $k \geq 10^6 \lambda^8 \Delta^{-1} \sqrt{n \log n} \cdot (\sqrt{\log n} + \Delta^{-\epsilon})$ is satisfied, and in that case, the total number of vertices selected by procedure Find is less than $n/3$.

In order to satisfy Property 1, we make a small change in procedure Find$_2$: During the calls to procedure Find$_2$ by algorithm Solve$_3$, the set U_2 is chosen only once, at the first call to Find$_2$, and its vertices are marked (the marks on the vertices are maintained globally in the algorithm). In the next calls to Find$_2$, we use the set U_2 that was chosen in the first call, without the vertices that were removed from the graph G. Moreover, the set U_1 is chosen randomly from all the unmarked vertices of V', and then its vertices are marked. Finally, the other random sets in the algorithm are chosen from the unmarked vertices of V', but their vertices are not marked. Clearly, the sets U_1 that are chosen at the different calls to Find$_2$ are disjoint, and therefore Property 1 is satisfied (the important pairs are all pairs (u, v) where $u \in U_2$ and $v \in U_1$ for some set U_1). For simplicity, we will analyze below the original procedure Find$_2$.

We now analyze procedure Find$_2$. We assume w.l.o.g. that the vertex u chosen in Step 3 belongs to the cluster V_1. Recall that $A_i = |V_i|$ and $A_{\max} = \max_i A_i$.

Lemma 6.1. If $k \geq 1200 \log n$ then with probability $1 - O(1/n^3)$, $A_1 \geq A_{\max} - 150\Delta^{-1}\sqrt{n \log n}$.

Proof. Let V_i be a cluster of maximum size. Denote $A'_j = |U_2 \cap V_j|$ for all j. Consider the following failure events:

1. $U_1 \cap V_i = \emptyset$.

2. $|d_{U_2}(v) - E[d_{U_2}(v)]| > \sqrt{3 \cdot 4u_2 \log n} = \frac{\sqrt{3}}{5}\sqrt{n \log n}$ for some $v \in U_1$.

3. $|A'_j - \frac{A_j}{n} u_2| > \sqrt{3 \cdot 4 \cdot u_2 \log n} = \frac{\sqrt{3}}{5}\sqrt{n \log n}$ for some j.

The probability of event 1 is at most $(1 - a_i)^{u_1} \leq e^{-a_i u_1} \leq e^{-u_1/m} = n^{-3}$. By Corollary 2.3 and Corollary 2.4, the probability that events 2 or 3 happen is $O(n^{-3})$. For the rest of the proof, assume that events 1–3 do not happen.

Let w be a vertex from $U_1 \cap V_i$. For $v \in V_j$ we have $E[d_{U_2}(v)] = A'_j\Delta + u_2 r$. Thus,

$$A'_1\Delta + u_2 r + \frac{\sqrt{3}}{5}\sqrt{n \log n} \geq d_{U_2}(u) \geq d_{U_2}(w) \geq A'_i\Delta + u_2 r - \frac{\sqrt{3}}{5}\sqrt{n \log n},$$

so $A'_1 \geq A'_i - \frac{2\sqrt{3}}{5}\Delta^{-1}\sqrt{n \log n}$. Therefore,

$$\begin{aligned}
A_1 &\geq \frac{n}{u_2}A'_1 - \frac{n}{u_2}\frac{\sqrt{3}}{5}\sqrt{n \log n} \\
&\geq \frac{n}{u_2}A'_i - 100 \cdot \frac{2\sqrt{3}}{5}\Delta^{-1}\sqrt{n \log n} - 100 \cdot \frac{\sqrt{3}}{5}\sqrt{n \log n} \\
&\geq A_i - 150\Delta^{-1}\sqrt{n \log n}.
\end{aligned}$$ ■

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. We again assume that $m \geq 3$. Define $d = \frac{1}{\lambda^2} \cdot a_1\Delta$. The events which can cause a failure of procedure Find$_2$ are:

1. $A_1 < A_{\max} - 150\Delta^{-1}\sqrt{n \log n}$.
2. $S_t \cap V_i = \emptyset$ for some i and t.
3. $|f_t(v) - E[f_t(v)]| > \frac{1}{5\lambda^2}D$ for some t and $v \in S_t$.
4. $|b_i^t - c_i^t| > d$ for some i and t.

Lemma 6.2. Let $\epsilon > 0$ be some constant. If $\max_i a_i < 2/3$, $k \geq 10^6\lambda^8\Delta^{-1}\sqrt{n \log n} \cdot (\sqrt{\log n} + \Delta^{-\epsilon})$, $p, r \in [\frac{1}{4}, \frac{3}{4}]$, and $\Delta \leq \frac{1}{10\lambda^3}$, then the probability that at least one of events 1–4 happens is $O(n^{-3})$.

Proof. The probability that event 1 happens is $O(n^{-3})$ (Lemma 6.1). By Theorem 4.1 we have that the probability that $|b_i^t - c_i^t| > 72a_{\max}\sqrt{\frac{m}{l}\log n}$ for some i and t is $o(n^{-3})$. Assuming that event 1 does not happen, $A_1 \geq A_{\max} - 150\Delta^{-1}\sqrt{n \log n} \geq (1 - 150/(10^6\lambda^8\sqrt{\log n}))A_{\max} = (1 - o(1))A_{\max}$. Therefore,

$$72a_{\max}\sqrt{\frac{m}{l}\log n} \leq \frac{72}{100\lambda^2} \cdot a_{\max}\Delta \leq \frac{1}{\lambda^2} \cdot a_1\Delta,$$

so the probability that event 2 happens (assuming that event 1 does not happen) is $o(n^{-3})$. The probability that event 3 or 4 happen is $O(n^{-3})$. ■

For the rest of the section assume that events 1–4 do not happen.

Lemma 6.3. Under the conditions of Lemma 6.2, with probability $1 - O(1/n^3)$,

1. $b_1^0, \hat{b}_1^0 \geq \frac{1}{2}a_1\Delta$.
2. For all $i > 1$, $-\frac{3}{2} \leq r_i^0, \hat{r}_i^0 \leq \frac{1}{2}$.
3. For all $1 \leq t \leq t^*$, $b_1^t \geq 2\Delta^{-\epsilon t} \cdot b_1^0$ and $\hat{b}_1^t \geq 2\Delta^{-\epsilon t} \cdot \hat{b}_1^0$.
4. For all $1 \leq t \leq t^*$ and all $i > 1$, $-2^{3^t} \leq r_i^t, \hat{r}_i^t \leq 1 - 1/2^{3^t}$.

Proof. Again, we prove the bounds of the lemma only for b_i^t and r_i^t, as the bounds for $\hat{b}_i^t$ and $\hat{r}_i^t$ are similar.

(1) By Lemma 4.2, we have that

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

Therefore, $b_1^0 \geq c_1^0 - d_1 \geq \left(\frac{2}{3} - \frac{1}{\lambda^2}\right)a_1\Delta > \frac{1}{2}a_1\Delta$.

(2) By Lemma 4.2, $c_i^0 = -2a_1a_i\Delta$ for every $i > 0$. Therefore, $b_i^0 \leq c_i^0 + d \leq d < \frac{1}{2}a_1\Delta$. Furthermore, $c_i^0 \geq -c_1^0$ (as $a_i \leq 1 - a_1$), so $b_i^0 \geq c_i^0 - d > -(1+\frac{1}{2})c_1^0$. Thus, $-\frac{3}{2} \leq r_i^0 \leq \frac{1}{2}$.

(3) Fix some t. We will show that $b_1^{t+1} \geq 2\Delta^{-\epsilon}b_1^t$. W.l.o.g. assume that $b_1^t \geq b_2^t \geq \cdots \geq b_m^t$.

As in the proof of Lemma 5.4, we have that

$$\begin{aligned}|b_i^t - b_j^t| &\leq |\hat{b}_i^t - \hat{b}_j^t| + 4\cdot\frac{1}{\lambda^2}\cdot a_1\Delta \leq \lambda\left(\lambda+1+\frac{8}{\lambda^2}\right)(\hat{b}_1^t - \hat{b}_2^t)\\ &\leq 2\lambda^2\cdot\frac{2D}{\Delta\hat{l}} = 200\lambda^4\cdot\frac{\sqrt{\log n}}{\Delta\sqrt{\hat{l}}} = \frac{1}{5\lambda^{1.5}}\cdot\frac{1}{\Delta\sqrt{l}}\end{aligned}$$

for all i and j.

Denote $F_0 = \gamma\Delta\sqrt{l}$ (where γ is defined in Lemma 4.3), $F = F_0a_{\max}$ and $Y = (\sum_{j=1}^m a_jb_j^t)/b_1^t$. By Lemma 4.3,

$$\begin{aligned}b_1^{t+1} \geq F_0a_1&\left(b_1^t - \sum_{j=1}^m a_jb_j^t - \left(2\Gamma\Delta - 5\sqrt{\frac{a_{\max}}{l}\log n}\cdot\Delta - \frac{2}{9}\left(\frac{1}{5\lambda^{1.5}}\right)^2\right)(b_1^t - b_m^t)\right)\\ &- 3\frac{a_1}{\sqrt{l}} - d_1.\end{aligned}$$

We have

- $(1-o(1))F \leq F_0a_1 \leq F$.
- $2\Gamma\Delta \leq \frac{1}{5\lambda^3}$.
- $5\sqrt{\frac{a_{\max}}{l}\log n}\cdot\Delta \leq \frac{5}{10^2\lambda^2}\cdot\frac{1}{10\lambda^3}$.
- $b_1^t - b_m^t \leq (1+\lambda)b_1^t$.
- $3\frac{a_1}{\sqrt{l}} = O\left(a_1\frac{\Delta}{\sqrt{m\log n}}\right) = o(Fb_1^t)$ (since $F \geq \frac{3}{2}\Delta\sqrt{l}/m \geq 150\lambda^2\Delta^{-\epsilon}$ and $b_1^t \geq \frac{1}{2}a_1\Delta$).
- $d_1 = \frac{1}{\lambda^2}\cdot a_1\Delta \leq \frac{1}{75\lambda^4}Fb_1^t$.

Thus,

$$\begin{aligned}b_1^{t+1} &\geq F_0a_1\cdot(1-Y)b_1^t - \left(\frac{\lambda+1}{5\lambda^3} + \frac{\lambda+1}{200\lambda^5} + \frac{2(\lambda+1)}{225\lambda^3} + \frac{1}{75\lambda^4} + o(1)\right)Fb_1^t\\ &\geq F_0a_1(1-Y)b_1^t - \frac{1}{4\lambda^2}Fb_1^t.\end{aligned}$$

Now,

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

Therefore, $1 - Y \ge \frac{1}{3}(1 - r_2^t) b_1^t \ge \frac{1}{3\lambda}$ and

$$\begin{aligned} b_1^{t+1} &\ge (1-o(1))(1-Y)Fb_1^t - \frac{1}{4\lambda^2} F b_1^t \ge \left(1 - o(1) - \frac{3\lambda}{4\lambda^2}\right)(1-Y)Fb_1^t \\ &\ge \left(1 - \frac{1}{\lambda}\right)(1-Y)Fb_1^t \ge \frac{1}{2} \cdot \frac{1}{3\lambda} \cdot F b_1^t \ge \frac{1}{6\lambda} \cdot 150\lambda^2 \Delta^{-\epsilon} b_1^t > 2\Delta^{-\epsilon} b_1^t. \end{aligned}$$

(4) By Lemma 4.2,

$$\begin{aligned} b_i^{t+1} \le F_0 a_i &\left(b_i^t - \sum_{j=1}^{m} a_j b_j^t + \left(2\Gamma\Delta - 5\sqrt{\frac{a_{\max}}{l} \log n} \cdot \Delta - \frac{2}{9}\left(\frac{1}{5\lambda^{1.5}}\right)^2 \right) (b_1^t - b_m^t) \right) \\ &+ 3\frac{a_i}{\sqrt{l}} + d \\ \le F_0 a_i &\left(b_i^t - \sum_{j=1}^{m} a_j b_j^t \right) + \frac{1}{4\lambda^2} F b_1^t \\ \le F &\max(0, b_i^t - \sum_{j=1}^{m} a_j b_j^t) + \frac{1}{4\lambda^2} F b_1^t \\ \le &\left(\max(0, r_i^t - Y) + \frac{1}{4\lambda^2} \right) F b_1^t. \end{aligned}$$

We have already shown above that

$$b_1^{t+1} \ge \left(1 - \frac{1}{\lambda}\right)(1-Y)Fb_1^t$$

Hence

$$r_i^{t+1} \le \frac{\max(0, r_i^t - Y) + \frac{1}{4}\lambda^{-2}}{(1-\lambda^{-1})(1-Y)} \le \left(1 + \frac{2}{\lambda}\right) \frac{\max(0, r_i^t - Y) + \frac{1}{4}\lambda^{-2}}{1 - Y}.$$

As

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

we have that $Y \ge r_m^t \ge -2^{3^t}$, and it follows that

$$\begin{aligned} r_i^{t+1} &\le \left(1 + \frac{2}{\lambda}\right) \frac{r_i^t - r_m^t + \frac{1}{4}\lambda^{-2}}{1 - r_m^t} \le \left(1 + \frac{2}{\lambda}\right) \frac{1 - \frac{1}{2^{3^t}} + 2^{3^t} + \frac{1}{4}\lambda^{-2}}{1 + 2^{3^t}} \\ &\le 1 + \frac{2}{\lambda} + \frac{2}{1 + 2^{3^t}} \cdot \frac{1}{4\lambda^2} - \frac{1}{2^{3^t}(1 + 2^{3^t})} \le 1 - \frac{1}{2^{3^{t+1}}}. \end{aligned}$$

Using similar arguments we obtain that

$$\begin{aligned} r_i^{t+1} &\geq \left(1+\frac{2}{\lambda}\right)\frac{\min(0, r_i^t - Y) - \frac{1}{4}\lambda^{-2}}{1-Y} \geq \left(1+\frac{2}{\lambda}\right)\frac{r_i^t - 1 - \frac{1}{4}\lambda^{-2}}{1-(\frac{2}{3}+\frac{1}{3}r_2^t)} \\ &= \left(1+\frac{2}{\lambda}\right)\cdot 3 \cdot \frac{r_i^t - 1 - \frac{1}{4}\lambda^{-2}}{1-r_2^t} \geq 3\left(1+\frac{2}{\lambda}\right)\frac{-\left(2^{3^t}+1\right) - \frac{1}{4}\lambda^{-2}}{1/2^{3^t}} \\ &\geq -4\cdot 2^{2\cdot 3^t} \geq -2^{3^{t+1}}. \end{aligned}$$

■

Lemma 6.4. Under the conditions of Lemma 6.2, with probability $1 - O(1/n^3)$, procedure Find$_2$ returns a subcluster of V_1 of size at least $1200\log n/\Delta^2$. 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. If $\Delta \leq \frac{1}{10\lambda^3}$, then from Lemma 6.3 we conclude (similarly to the proof of Lemma 5.6) that with probability $1-O(1/n^3)$, procedure Find$_2$ returns a subcluster of size at least $1200\log n/\Delta^2$.

We now consider the case when $\Delta > \frac{1}{10\lambda^3}$. We will show that procedure Find$_2$ stop after the first iteration and return a subcluster of V_1 of the required size.

W.l.o.g. assume that $\hat{b}_1^0 \geq \hat{b}_2^0 \geq \cdots \geq \hat{b}_m^0$. The proofs of items 1 and 2 in Lemma 6.3 are also valid for the case $\Delta > \frac{1}{10\lambda^3}$. Therefore, we have that $\hat{b}_1^0 - \hat{b}_2^0 \geq \frac{1}{2}\hat{b}_1^0 \geq \frac{1}{4}a_1\Delta \geq \frac{1}{5}\frac{\Delta}{m}$. For $v \in V_i$, $E\left[f_0(v)\right] = \Delta\hat{l}\hat{b}_i^0$. Thus,

$$\min_{v\in S_0\cap V_1} f_0(v) - \max_{v\in S_0 - V_1} f_0(v) \geq \Delta\hat{l}(\hat{b}_1^0 - \hat{b}_2^0) - 2\cdot\frac{1}{5\lambda^2}D \geq \frac{1}{10\lambda^3}\cdot\hat{l}\cdot\frac{1}{5}\frac{\Delta}{m} - \frac{2}{5\lambda^2}D > D.$$

Hence, procedure Find$_2$ stops at Step 10 and $\{v_1,\ldots,v_j\} = V_1\cap S_0$.

As the number of iterations is at most $\lceil 1/\epsilon\rceil = O(1)$, the time complexity of Find$_2$ is $O(s\hat{l} + l\hat{l} + u_1u_2) = O(l\hat{l} + u_1u_2) = O(m^4\Delta^{-4(1+\epsilon)}\log n + m^2\Delta^{-4}\log^3 n + mn\log n)$. ■

Combining Lemma 5.6 and Lemma 6.4 proves Theorem 1.2.

Acknowledgements

We thank Noga Alon, Chaim Linhart, and Yossi Richter for helpful discussions.

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** (1999), 281–297.

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

[3] T. Carson and R. Impagliazzo, *Hill-climbing finds random planted bisections*, Proc. 12th Symposium on Discrete Algorithms (SODA 01), ACM press, 2001, pp. 903–909.

[4] A. E. Condon and R. M. Karp, *Algorithms for graph partitioning on the planted partition model*, Random Structures and Algorithms **18** (2001), no. 2, 116–140.

[5] M. E. Dyer and A. M. Frieze, *The solution of some random NP-hard problems in polynomial expected time*, J. of Algorithms **10** (1989), no. 4, 451–489.

[6] U. Feige and J. Kilian, *Heuristics for semirandom graph problems*, J. of Computer and System Sciences **63** (2001), no. 4, 639–671.

[7] W. Hoeffding, *Probability inequalities for sums of bounded random variables*, J. Amer. Statist. Assoc. **58** (1963), 13–30.

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

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

[10] C. McDiarmid, *Centering sequences with bounded differences*, Combinatorics, Probability and Computing **6** (1997), no. 1, 79–86.

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

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

[13] P. van Beek, Z. Wahrscheinlichkeitstheorie verw. Geb. **23** (1972), no. 3, 187–196.

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

List of Symbols

Symbol	Meaning	Introduced on page
Δ	$p - r$	1
Γ	$\max_i \lvert a_i - \frac{1}{m} \rvert$	17
A_i	$\lvert V_i \rvert$	7
a_i	$\lvert V_i \rvert / n$	7
$A_{\max}$	$\max_i A_i$	7
$a_{\max}$	$A_{\max}/n$	7
b_i^t	$\mathrm{I}(V_i, L_t, R_t)$	23
$c_i(f)$	$2a_i \sum_{j \neq i} a_j (p_{ij}^{>}(f) - p_{ij}^{<}(f))$	14
c_i^0	$c_i(d_{\{u\}})$	23
c_i^t	$c_i(d_{L_t} - d_{R_t})$	23
$d_S(v)$	Number of neighbors of v in S	3
d_t	$\begin{cases} \frac{1}{100} \cdot \frac{\Delta}{m} & \text{if } t \leq 2 \\ \frac{1}{100} \cdot \frac{\Delta}{m} \sqrt{\log n} & \text{if } t \geq 3 \end{cases}$	23
$f_t(v)$	$d_{\hat{L}_t}(v) - d_{\hat{R}_t}(v)$	23
$\mathrm{I}(V_i, L, R)$	$\frac{\lvert V_i \cap L \rvert - \lvert V_i \cap R \rvert}{\lvert L \rvert}$	13
k	$\min_i \lvert V_i \rvert$	1
m	Number of clusters	1
n	Number of vertices	2
p	Probability for an edge between vertices of same cluster	1
$p_{ij}^{>}(f)$	$P\left[f(v) > f(w) \mid v \in V_i \cap T, w \in V_j \cap T\right]$	14
r	Probability for an edge between vertices of different clusters	1
r_i^t	b_i^t / b_1^t	23
V_i	i-th underlying cluster	1