Thesis for the degree
Master of Science

Submitted to the Scientific Council of the
Weizmann Institute of Science
Rehovot, Israel

עבודת גמר (תזה) לתואר
מוסמך למדעים

מוגשת למועצה המדעית של
מכון ויצמן למדע
רחובות, ישראל

By
Noa Dina Loewenthal

מאת
נועה דינה לוונטל

ניצול מידע טמפורלי לאיתור קהילות ברשתות חברתיות
Exploiting temporal information for detecting communities in social networks

Advisor: Prof. Robert Krouthgamer

מנחה: פרופ' רוברט קראוטגמר

November 2014

חשוון התשע"ה

Dedicated to Hannah and Manfred, Leah and Yehuda,
my grandparents who left me a legacy of education

Abstract

Community detection is a major problem in the study of social networks. Information regarding the creation process of the graph, which is temporal in nature, may potentially be useful for algorithms trying to find communities. We wish to examine if such temporal information can indeed be leveraged algorithmically and if so, how much temporal information is needed to achieve improvement.

We model a community in a social network as a clique in a graph, and temporal information as edge labels that describe the time in which the edge was added to the graph. In our model, edges inside the community are added gradually, simulating a community in which members are added one by one while edges outside the community are added randomly (one can think of the community edges as signal and of other edges as noise). In our setup, when no temporal information is available, finding the clique requires solving the NP-hard Maximum-Clique problem or the Hidden Clique Problem.

We model access to the temporal information in two different ways, one is by querying the time in which an edge was added, the second is by querying a step and getting all the edges that were added to the graph at that step. For each access method we design algorithms that can find the community, even if only limited portions of the temporal information are available. Overall, we see that exploiting temporal information allows polynomial time algorithm to the problem which otherwise have no known efficient solutions.

Acknowledgments

I would like to express my sincere gratitude to my advisor, Prof. Robert Krauthgamer, for his continuous support, motivation, enthusiasm and patience. His excellent guidance and ideas made a significant contribution both to this work and to my personal research experience.

My gratitude also goes to all the faculty, staff and my fellow students here at the Weizmann Institute of Science that provided the atmosphere and the environment for such work.

Finally, I thank my family - my parents and brothers and my significance other, for being loving and encouraging throughout this enriching journey.

Contents

1 Introduction **6**
- 1.1 Motivation - temporal information in social networks . . . 6
- 1.2 Maximum-Clique and Hidden Clique Problems . . . 6
- 1.3 Our model - clique with temporal information . . . 7
- 1.4 Our results . . . 8
  - 1.4.1 The worst case model . . . 8
  - 1.4.2 Hidden clique model . . . 8

2 Worst-case graphs with random time stamps **10**
- 2.1 The model . . . 10
- 2.2 Uniqueness of K . . . 10
- 2.3 Finding the clique . . . 11

3 Hidden clique problem with edge queries **14**
- 3.1 The model . . . 14
- 3.2 Preliminary observations . . . 14
- 3.3 Upper bound . . . 15
  - 3.3.1 Analysis of steps 7-14 - construction of Q . . . 17
  - 3.3.2 Analysis of steps 16-18 - construction of $\Gamma_\cap$ . . . 17
  - 3.3.3 Analysis of steps 19-20- constriction of C . . . 18
  - 3.3.4 Success probability . . . 19
  - 3.3.5 Number of queries and running time . . . 19
  - 3.3.6 Procedure $\textsc{Test}(G, Q)$ . . . 20
- 3.4 Lower bound . . . 24
  - 3.4.1 Extension to randomized algorithms . . . 28

4 Hidden clique problem in snapshots model **30**
- 4.1 Algorithm that uses one query . . . 30
- 4.2 Algorithm that uses many queries . . . 33

A Additional proofs for section 3 **39**

1 Introduction

1.1 Motivation - temporal information in social networks

Many graphs derived from real world problems develop over time, but many algorithmic approaches for analyzing these graphs, rely only on the resulting graph and ignore the graphs' creation process. The use of this additional information may provide crucial information for solving problems on graphs and improving the time complexity of algorithms

In light of the growth of social networks, we focus on the problem of finding communities in a social network, e.g., the graph of friendships in Facebook or Twitter and the World Wide Web (as a generalized social network). We present two models of how a social network and a community are constructed over time and explain how algorithms may exploit the temporal information - the time at which edges (friendship in the social network) are introduced to the network. We then design algorithms that find a community in such a network in different settings. We model a social network by a graph and a community by a clique in a graph, which we discuss next.

1.2 Maximum-Clique and Hidden Clique Problems

A clique in a graph $G = (V, E)$ is a set of vertices such that every two are connected by an edge. Let $\omega(G)$ denote the size of the largest clique in G. The problem of computing $\omega(G)$ in an input graph G is called the Maximum-Clique problem and is NP-hard [Kar72]. The best approximation known for this problem is an $O(\frac{n(\log\log n)^2}{\log^3 n})$ approximation [Fei04]. In fact, for every fixed $\epsilon > 0$ it is impossible to approximate $\omega(G)$ within a ratio of $n^{1-\epsilon}$ in polynomial time assuming NP does not have a polynomial time randomized algorithm (i.e. NP$\neq$ZPP)[Hs99].

The hardness of approximating the Maximum-Clique problem in the worst case suggests studying the average case. Let $G(n, \frac{1}{2})$ denote a random graph on n vertices such that each possible edge appears independently with probability $\frac{1}{2}$. It is well known that in this model, with probability tending to 1 as n tends to infinity, $\omega(G) = 2\log n + o(\log^* n)$ [GM75]. There are a few polynomial algorithms that find with high probability a clique of size $(1+o(1))\log n$ in $G(n, \frac{1}{2})$ (e.g.[GM75]), i.e. a clique of about half the size of the largest clique. There is no known algorithm for finding in this $G(n, \frac{1}{2})$ a clique of size $(1+\epsilon)\log n$ for any fixed $\epsilon > 0$, which is an open question dating back to [Kar76].

A natural possible direction to attack these questions is looking at similar random models but assure that the largest clique is larger than $2\log n$. This intuition leads to another well known and challenging problem - the Hidden Clique problem that was suggested by [Jer92] and [Kuč95]. In this problem first a graph G is drawn from the distribution $G(n, \frac{1}{2})$, then a set K of k nodes is chosen uniformly at random and every two nodes in K are connected by an edge, making K a clique, the goal is to find the largest clique. We denote by $G(n, \frac{1}{2}, k)$ the resulting distribution of graphs. It is well known that for $k \gg \log n$ with high probability the largest clique is exactly the planted clique K. For the case $k \geq \Omega(\sqrt{n})$ polynomial-time algorithms with high success probabilities (taken over the input distribution) for the

hidden clique problem are presented in [AKS98, AV11, DGGP14, DM13, FK00, FR10]. A new approach to solve hidden clique for $k \leq n^{\frac{1}{2}}$, if a certain "generalization of eigenvalue computation" can be computed in polynomial time (even approximately) is suggested in [FK08, BV09]. Along with [FK03, Jer92] that show that two possible approaches to the problem fail for $k = o(n^{\frac{1}{2}})$,we get that $k = \Omega(n^{\frac{1}{2}})$ seems like a natural barrier for the problem.

1.3 Our model - clique with temporal information

We model the problem of finding a community in a social network, as finding a clique in a graph. The edges of the graph are added over time (in discrete steps), information regarding this process of construction is referred to as temporal information. In our models, edges inside the community are added in an ordered fashion, simulating members joining the community one by one (representing signal), while edges outside the community are added in random steps (representing noise). The models are such that given no information about the construction process (only the resulting network) the problem is either the (worst-case) Maximum Clique problem or the (average-case) Hidden Clique problem.

Formally, we present three models, in all of them the graph $G = (V, E)$ with $|V| = n$ has a clique of size k denoted by $K = \{w_1, \ldots, w_k\}$, the graph is constructed over k discrete steps such that in each step a node is added to the clique and edges are only added to the graph.

When considering the graph without temporal information, we think of the result of the construction process without the information regarding the step in which each edge was added to it. For every pair $u, v \in V$ we denote by $\tau(u, v)$ the step in which they were connected ($\tau(u, v) = 0$ if $(u, v) \notin E$). Temporal information (regarding the construction of G) is accessible in each model in one of two ways:

- Edge query - by querying the step in which a pair of nodes $u, v \in E$ were connected (the answer is 0 if $(u, v) \notin E$).
- Snapshot query - by querying the set E_t of all the edges in G that were added in step $1 \leq t \leq k$.

Note that these two access methods provide the same information overall because given all edge queries one can derive all the snapshots, and given all snapshots one can derive the answer to any edge query.

We examine three models that differ in three parameters - the underlying graph G, the construction process of the graph over time and the access method to the information. Informally, the three models are:

- A worst-case graph with edge queries.
- A hidden clique graph with:
  - edge queries.
  - snapshot queries.

1.4 Our results

In all three models we present polynomial time algorithms that find the clique K with high probability. In fact, our algorithms use only a small portion of the temporal information. We conclude that the availability of temporal information, even in limited form and amount, yields a significant improvement in the computational complexity of finding a community in a social network. Putting to work this concept of exploiting temporal information in real-world social networks may be an interesting though more empirical future work. In light of our theoretical results that even a small amount of temporal information suffices, it may also be useful to quantify how much information is needed in such real networks.

1.4.1 The worst case model

In the worst case model, the only assumption we make regarding the graph G is that it has at least one clique of size k, let $K = \{w_1, \ldots, w_k\}$ be one such clique. The construction process (that is not available to the algorithm) of G is different inside K and outside it:

- For $w_i, w_j \in K$ the edge (w_i, w_j) is added to the graph at step $\min\{i, j\}$, this models nodes that join the community in order $w_1, \ldots, w_k$.
- For $(u, v) \in E$ such that at least one of u, v is not in K, the step in which the edge (u, v) is added to G is chosen uniformly at random from $[k]$.

In this model, access to the information is via edge queries i.e. upon querying u, v we get $\tau(u, v)$.

Finding a clique of size k in G (without further information) is NP-hard. One might think that additional temporal information regarding the construction of the graph might not suffice to find K with high probability. We show that this is not the case when $k = \Omega(\log n)$.

Theorem 1.1. *There exists a polynomial-time algorithm with the following guarantees: Given as input k and a graph G that has a clique K of size $k = \Omega(\log n)$ together with time-stamps that are generated according to the above, with probability at least $1 - \frac{2}{n}$ over the time-stamps the algorithm outputs K.*

1.4.2 Hidden clique model

In the hidden clique model an input graph $G = (V, E)$ with a community $K = \{w_1, \ldots w_k\}$ is constructed over k steps. The set of edge E is composed of E_t for $1 \leq t \leq k$, where E_t are exactly the edges added at step t, i.e. $E = \cup E_t$. Each E_t consists of edges of two types that simulate the different behavior inside K and outside it:

- Random edges: each pair of (distinct) vertices that has not been connected in any $E_{t'}$ for $t' < t$ is connected in E_t with probability $p = 1 - (\frac{1}{2})^{\frac{1}{k}}$ independently of all other events. Note that this includes also edges inside K.

- Clique edges: edges between w_t and any other $w_i \in K$ (that it is not connected to yet) are added with probability 1, namely $\{(w_t, w_i) \mid i > t \text{ and } \forall t' < t\ (w_t, w_i) \notin E_{t'}\}$, this models nodes that join the community in order $w_1, \ldots, w_k$.

This constructions creates graphs that are taken from the distribution $G(n, \frac{1}{2}, k)$.

We note that the distribution of time stamps in this model differs from its distribution in the model for worst case graphs. We have made this change to test the idea of temporal information on different models and make sure it is not sensitive to the specific model of temporal information.

In this model of graph construction we use two ways for accessing the information - edge queries and snapshot queries. In both ways of accessing the information we show that the known barrier of $k = \Omega(n^{\frac{1}{2}})$ can be broken, even when only part of the temporal information is available.

In the edge query model we show that even if only part of the information is available then a planted clique of polynomial size (i.e. $k = n^\alpha$ where $\alpha > 0$ is a constant) can be found efficiently.

Theorem 1.2. *Let $c = 0.99$ (although any constant $0 < c < 1$ would work). Assuming $k = n^\alpha$ for constant $0 < \alpha < \frac{1}{2}$, there exists an efficient algorithm that gets G, k as input, uses $q = \Theta(n^{2-c\alpha})$ edge queries and with probability at least $1 - \frac{1}{n}$ finds the hidden clique.*

On the other hand, we show that assuming that the hidden clique problem does not have a polynomial algorithm then $\log n$ edge queries do not suffice.

Theorem 1.3. *Suppose there exists a randomized polynomial time algorithm A that gets G as input and uses $s = O(\log n)$ adaptive edge queries to solve the hidden clique problem in the edge queries model with high probability. Then there exists a probabilistic polynomial time algorithm B that solves the hidden clique problem in the classic model (i.e. with no edge queries) with at least the same probability.*

In the snapshot model we show that even given only one snapshot the barrier of $k = \Omega(n^{\frac{1}{2}})$ is broken and K can be found for $k = \widetilde{\Omega}(n^{\frac{1}{3}})$.

Theorem 1.4. *Assume $k \geq cn^{\frac{1}{3}} \ln n$ for large enough constant $c > 0$ and fix a constant $0 < \beta < 1$. For any $t \leq \beta k$, given G and E_t one can find K in polynomial time with high probability.*

The range of values of k for which K can be found in polynomial time in this model can be further expanded. Given $\Omega(\log n)$ snapshots, a clique of polynomial size (i.e. $k = n^\alpha$ where $\alpha > 0$ is a constant) can be found.

Theorem 1.5. *Assuming $|K| = k = n^\alpha$ for a fixed $\alpha > 0$, there exists an algorithm that runs in $n^{O(\frac{1}{\alpha})}$ time and given $\{E_t \mid t \in I\}$ and I for any set $I \subseteq [k]$ of size $\geq a \ln n$ for $a \geq c \cdot 2^{\frac{10}{\alpha}+2}$ and large enough constant c, finds K with probability at least $1 - \frac{5r}{n}$ (where $r = r(\alpha, a)$ is a parameter to be determined later that does not depend on n).*

2 Worst-case graphs with random time stamps

2.1 The model

Let $G = (V, E)$ be an arbitrary graph that has a clique of size k. Let $K = \{w_1, \ldots, w_k\}$ be a clique of size k in G. Time-stamps are generated for the edges of the graph as follows:

- For an edge $e = (w_i, w_j)$ let $\tau(e) = \min\{i, j\}$.
- For an edge $e = (u, v)$ such that $|\{u, v\} \cap K| \leq 1$ choose $\tau(e)$ uniformly at random from $[k]$.

2.2 Uniqueness of K

We note that in this model, the labels of the edges inside K have a specific structure - for every $i \in [k]$, there are exactly $k - i$ edges with the label i and they are all incident to the same node w_i, we call this structure a "correct structure". Here we show that with high probability, there is no other set of nodes of size k that has a correct structure.

Claim 2.1. *Assuming $k = \Omega(\log n)$, with probability at least $1 - \frac{1}{n}$, the only set of size k that has a correct structure is K.*

The assumption that k is large is necessary, and without it the theorem is not true. For instance, for $k = O(1)$ each clique of size k has a constant probability to have a correct structure and so we expect to see many such sets besides K (for a graph G with many k-cliques).

Proof of Claim 2.1. Let $m = \lceil \log_k n \rceil$ be the smallest integer such that $k^m \geq n$.

Throughout this proof, whenever considering a set of nodes S, we assume that it is a clique. Clearly, this is not the case for all sets of nodes (unless $G = K_n$) thus, the probability to have a clique $C \neq K$ of size k with a correct structure is at most the one presented here.

Consider a set S of nodes that forms a clique. Assume that S has $1 \leq j \leq k$ nodes outside K and $k - j$ nodes in common with K. We wish to bound the probability that this set has an internal structure of time-stamps as K. The edges between the $k - j$ nodes that are part of K are labeled correctly, so we need to bound the probability that the other edges are labeled to fit this. Denote by $\{w_{t_1}, \ldots, w_{t_{k-j}}\} = S \cap K$ and assume that $t_1 < t_2 < \cdots < t_{k-j}$.

We note that by looking at the labels of the subgraph of G induced by $S \setminus K$ we can determine the names of $w_{t_1}, \ldots, w_{t_{k-j-2}}$, we will also know that $\tau(w_{t_{k-j-1}}, w_{t_{k-j}}) = t_{k-j-1}$ so we will know that one of $w_{t_{k-j-1}}, w_{t_{k-j}}$ is $w_{t_{k-j-1}}$ but we will not be able to determine which of the two nodes is it since both options create the same time-stamps structure. So we need to guess which one is $w_{t_{k-j-1}}$ and then we have $j+1$ names to give to the nodes $\{u\} \cup (S \setminus K)$ where $u \in \{w_{t_{k-j-1}}, w_{t_{k-j}}\}$ the node we guessed is not w_{k-t-1}. After choosing a "name" for j nodes in $S \setminus K$ plus one node in K, there is exactly one label for each edge that will fit the structure. So given a naming of the $j + 1$ nodes, the probability to get the desired structure is $(\frac{1}{k})^{j(k-j)+\binom{j}{2}}$.

There are at most $(j+1)!$ ways to give names to $\{u\} \cup (S \setminus K)$ (after guessing u) and there are 2 ways of guessing u, thus, the probability that a specific set S has a correct structure is at most $2(j+1)!(\frac{1}{k})^{j(k-j)+\binom{j}{2}}$.

For a fixed j, there are $\binom{n-k}{j}$ ways to choose the nodes of $S \setminus K$, and there are $\binom{k}{k-j}$ ways to choose the nodes of $S \cap K$, so overall there are $\binom{n-k}{j}\binom{k}{k-j}$ such possible sets S. Thus

$$\begin{aligned}
&\Pr[\exists v_1, \ldots v_j \text{ that together with a subset of } K \text{ form a correct structure}] \\
&\leq \binom{n-k}{j}\binom{k}{k-j} 2(j+1)!(\frac{1}{k})^{j(k-j)+\binom{j}{2}} \\
&\leq \frac{n^j}{j!} k^{k-j} j! k^2 (\frac{1}{k})^{j(k-j)+\binom{j}{2}} \\
&\leq k^{mj+k-j+2-j(k-j)-\binom{j}{2}} \\
&= k^{k(1-j)+mj+2+\frac{j}{2}(j-1)}.
\end{aligned}$$

It suffices to show that the latter is at most $k^{-2m} \leq n^{-2}$, thus it is enough to have

$$k(1-j) + mj + 2 + \frac{j}{2}(j-1) \leq -2m.$$

For $j > 1$, this is equivalent to

$$k \geq \frac{2m + mj + 2 + \frac{j}{2}(j-1)}{j-1} = m + \frac{3m+2}{j-1} + \frac{j}{2},$$

which holds whenever $k = \Omega(\log n)$ since in this case $m = O(\frac{\log n}{\log \log n})$ and $j \leq k$.

For the case $j = 1$ we get

$$\Pr[\exists v_1 \text{ that together with a subset of } K \text{ form a correct structure}] \leq \frac{2(n-k)k}{k^{k-1}} < \frac{1}{n^2}.$$

Altogether

$$\begin{aligned}
&\Pr[\exists 1 \leq j \leq k \text{ and } \{v_1, \ldots, v_j\} \text{ that together with a subset of } K \text{ form a correct structure}) \\
&\leq \frac{k}{n^2} \leq \frac{1}{n}.
\end{aligned}$$

□

2.3 Finding the clique

Theorem 1.1. *There exists an algorithm that gets as input k and a graph G that has a clique K of size $k = \Omega(\log n)$ together with time-stamps that are generated according to the above. The algorithm finds K with probability at least $1 - \frac{2}{n}$ over the time-stamps and runs in time $n^{O(m)}(3m)!$ where $m = \lceil \log_n k \rceil$.*

We note that the algorithm presented in this part uses graph properties such as adjacency and time-stamps but does not use the names of the nodes themself hence the correctness presented holds for any permutation of the nodes' names.

Note: The probability of success is taken over the time-stamps, i.e. for every G the algorithm succeeds with this probability.

Proof of Theorem 1.1. Note that m is the smallest integer such that $k^m \geq n$.

For a tuple $\sigma = (v_1, \ldots, v_{3m}) \in V^{3m}$ define $U(\sigma) = \{u \in V \mid \tau(v_1, u) = 1, \ldots, \tau(v_{3m}, u) = 3m\}$. We note that for a given tuple σ, it takes $O(nm)$ time to find $U(\sigma)$.

Claim 2.2. $\Pr[K = \{w_1, \ldots, w_{3m}\} \cup U(w_1, \ldots, w_{3m})] \geq 1 - \frac{1}{n^2}$.

Proof of Claim 2.2. By the construction of the labels in the graph, it is easy to see that $K \setminus \{w_1, \ldots, w_{3m}\} \subseteq U(w_1, \ldots, w_{3m})$. We now show that with high probability $U(w_1, \ldots, w_{3m}) = K \setminus \{w_1, \ldots, w_{3m}\}$. For a node $u \in V \setminus K$

$$\Pr[u \in U(w_1, \ldots, w_{3m})] = \frac{1}{k^{3m}} \leq \frac{1}{n^3}.$$

By union bound we get that

$$\Pr[\exists u \in V \setminus K \text{ s.t. } u \in U(w_1, \ldots, w_{3m})]) \leq \frac{1}{n^2}.$$

So with high probability $K = \{w_1, \ldots, w_{3m}\} \cup U(w_1, \ldots, w_{3m})$. □

For a tuple $\sigma = (v_1, \ldots, v_{3m}) \in V^{3m}$, we denote by $S_\sigma = \{v_1, \ldots, v_{3m}\}$ the set of nodes that are in σ.

We now show that although $(w_1, \ldots, w_{3m})$ is not given to us, we can find K by an exhaustive search over all tuples of size $3m$. The intuition is that for a tuple $\sigma \neq (w_1, \ldots w_{3m})$, even if the set $S_\sigma \cup U(\sigma)$ is a clique of size k, by Claim 2.1 with high probability it does not have a correct structure and by Claim 2.2 $S_{(w_1, \ldots w_{3m})} \cup U(w_1, \ldots w_{3m}) = K$ with high probability and so it can be identified.

Algorithm WorstCaseAlgorithm (V, E, k)

1: **for** every tuple $\sigma = (v_1, \ldots, v_{3m}) \in V^{3m}$ **do**
2: **if** $\{v_1, \ldots, v_{3m}\} \cup U(\sigma)$ is a clique of size k that has a correct structure **then**
3: **return** $\{v_1, \ldots, v_{3m}\} \cup U(\sigma)$
4: **return** $\emptyset$

If the algorithm gets to the iteration in which $\sigma = (w_1, \ldots w_{3m})$ then with probability at least $1 - \frac{1}{n}$ it returns K since according to Claim 2.2 with high probability in this case $S_\sigma \cup U(\sigma) = K$ (and K meets the condition of step 2) so the correct answer will be returned.

For the algorithm to fail at least one of the following events must happen:

1. $\{w_1, \ldots, w_{3m}\} \cup U(w_1, \ldots, w_{3m}) \neq K$.

2. There exists $\sigma \neq (w_1, \ldots, w_{3m})$ such that $S_\sigma \cup U(\sigma)$ is a clique of size k that has a correct structure.

The probability that event 1 happens is according to Claim 2.2 at most $\frac{1}{n^2}$.

Lemma 2.3. Pr[*event 2 happens*] $\leq \frac{1}{n}$

Proof of Lemma 2.3. There are two possible cases:

1. $S_\sigma = \{w_1, \ldots, w_{3m}\}$ and $\sigma \neq (w_1, \ldots, w_{3m})$.

2. $S_\sigma \neq \{w_1, \ldots, w_{3m}\}$.

We show that in both cases $S_\sigma \cup U(\sigma) \neq K$ and so by Theorem 2.1 with probability at most $\frac{1}{n}$ there is a tuple σ such that $S_\sigma \cup U(\sigma)$ has a correct structure.

In the first case, there exists $1 \leq i \leq 3m$ such that $v_i = w_r \neq w_i$. For every $j > 3m$ $\tau(w_j, v_i) = r \neq i$ so $K \cap U(\sigma) = \emptyset$ and $S_\sigma \cup U(\sigma) \neq K$.

In the second case, there must exists $v_i \in S_\sigma$ such that $v_i \notin \{w_1, \ldots, w_{3m}\}$. If $v_i \notin K$ it is obvious that $S_\sigma \cup U(\sigma) \neq K$. Otherwise it must be that $v_i = w_t$ for some $t > 3m$, then for every $j \neq i$ $\tau(v_i, w_j) \neq i$ so $w_j \notin S(\sigma)$ and so $S_\sigma \cup U(\sigma) \neq K$. □

So the algorithm fails with probability at most $\frac{2}{n}$. It is easy to see that the running time of this algorithm is $n^{O(m)}(3m)!$ this concludes the proof of Theorem 1.1. □

3 Hidden clique problem with edge queries

3.1 The model

A graph $G = (V = [n], E)$ with a clique $K = \{w_1, ..., w_k\} \subseteq V$ of size $k = n^\alpha$ for some constant $0 < \alpha < 1$ is constructed over k steps. At each step $1 \leq t \leq k$ an edge set E_t is added and $E = \cup_{t=1}^{k} E_t$. The set E_t consists of edges of two types:

- Random edges: each pair of (distinct) vertices that is not already connected in any $E_{t'}$, $t' < t$, is connected in E_t with probability $p = 1 - (\frac{1}{2})^{\frac{1}{k}}$ independently of all other events. Note that this includes also edges inside K.
- Clique edges: edges between w_t and all other vertices in K (to which w_t is not already connected) are added with probability 1, namely $\{(w_t, w_i) | i \neq t\}$.

For $(u, v) \in E$ we denote by $\tau(u, v)$ the step in which the edge was added to G. If $(u, v) \notin E$ we define $\tau(u, v) = 0$. In this section we consider the edge query method for accessing the temporal information, i.e. the algorithm query $\tau(u, v)$ for a pair of nodes $u, v \in E$.

The above construction of G with the same temporal information (edge labels) will also be used in section 4, but there the temporal information is accessed via snapshots. We stress that these two access methods provide the same information overall, because given all edge queries one knows all the snapshots, and given all snapshots one knows the answer to any edge query. Our algorithms use only part of the temporal information, hence our results in theses sections are not equivalent.

3.2 Preliminary observations

Lemma 3.1. *$p = \Theta(\frac{1}{k})$, to be exact $\frac{\ln 2}{k + \ln 2} \leq p \leq \frac{\ln 2}{k}$.*

Proof of Lemma 3.1. We first show that $p = O(\frac{1}{k})$, we note that[1] $(1 + \frac{-\ln 2}{k})^k \leq e^{-\ln 2}$ and so $1 - (\frac{1}{e^{\ln 2}})^{\frac{1}{k}} \leq 1 - ((1 + \frac{-\ln 2}{k})^k)^{\frac{1}{k}}$ and

$$p = 1 - (\frac{1}{2})^{\frac{1}{k}} = 1 - (\frac{1}{e^{\ln 2}})^{\frac{1}{k}} \leq 1 - ((1 - \frac{\ln 2}{k})^k)^{\frac{1}{k}} = \frac{\ln 2}{k}.$$

So indeed $p = O(\frac{1}{k})$.

We now show that $p = \Omega(\frac{1}{k})$. First note that $e^{\ln 2} \geq (1 + \frac{\ln 2}{k})^k$, from this we have $(\frac{1}{e^{\ln 2}})^{\frac{1}{k}} \leq ((1 + \frac{\ln 2}{k})^{-k})^{\frac{1}{k}}$ and

$$p = 1 - (\frac{1}{2})^{\frac{1}{k}} = 1 - (\frac{1}{e^{\ln 2}})^{\frac{1}{k}} \geq 1 - ((1 + \frac{\ln 2}{k})^{-k})^{\frac{1}{k}} = \frac{\ln 2}{k + \ln 2}.$$

So indeed $p = \Omega(\frac{1}{k})$ and from the proof we can see that $\frac{\ln 2}{k + \ln 2} \leq p \leq \frac{\ln 2}{k}$. □

[1] For $x \in \mathbb{R}$ we have $1 + x \leq e^x$ we take $x = \frac{-\ln 2}{k}$ and raise it to power $k > 0$.

Lemma 3.2. *Outside the clique K, the graph looks like $G(n, \frac{1}{2})$.*

Proof of Lemma 3.2. For $v \neq u$ such that $u \notin K$ it holds that

$$\begin{aligned}\Pr[(u,v) \in E] &= \Pr[\exists 1 \leq t \leq k \text{ s.t. } (u,v) \in E_t] \\ &= 1 - \Pr[\forall 1 \leq t \leq k \text{ s.t. } (u,v) \notin E_t] = 1 - (1-p)^k = \frac{1}{2}.\end{aligned}$$

Moreover, this event is independent of all other edges, therefore outside K, the graph looks like $G(n, \frac{1}{2})$. □

Note that by Lemma 3.2, the resulting graph without the edge labels is an instance of the hidden clique problem with parameters $(G, \frac{1}{2}, n^\alpha)$, recall that for $\alpha < \frac{1}{2}$ there are no known polynomial algorithm to this problem. Since there are efficient algorithms that find K with high probability without additional information when $\alpha \geq \frac{1}{2}$, we focus on the case that $\alpha < \frac{1}{2}$.

3.3 Upper bound

We show that even without full temporal information the clique K can be found with high probability in polynomial time. In this section we think of K as built from three sets of nodes according to the order in which they are added to the graph $K = K^{1/3} \cup K^{2/3} \cup K^{3/3}$ where $K^{i/3} = \{w_{\frac{(i-1)k}{3}+1}, \ldots, w_{\frac{ik}{3}}\}$ for $1 \leq i \leq 3$.

Theorem 1.2. *Let $c = 0.99$ (although any constant $0 < c < 1$ would work). Assuming $0 < \alpha \leq \frac{1}{2}$ is a constant, there exists an efficient algorithm that gets G, k as input, uses $q = \Theta(n^{2-c\alpha})$ edge queries and finds the hidden clique with probability at least $1 - \frac{1}{n}$.*

Proof of Theorem 1.2. Let m' be the unique integer such that $\frac{1}{\alpha} - 2 < m' \leq \frac{1}{\alpha} - 1$.

We present algorithm Pick(G, α). It uses procedure Test that will be described later. Algorithm Pick first finds (using procedure Test) a set Q (line 7) which is with high probability a random subset of $K^{2/3}$ together with the names of the nodes of Q. The algorithm then proceeds to find all the nodes that are connected to all of Q with edges that are labeled by the name of the node from Q (line 18), this set is $\Gamma_\cap(Q)$. Since the nodes are added to K one by one, we expect that many nodes of $K^{3/3}$ will be in $\Gamma_\cap(Q)$, and since edges from nodes in K to nodes outside K have random labels, we do not expect to have any node from $V \setminus K$ in $\Gamma_\cap(Q)$; we conclude that $\Gamma_\cap(Q) \subseteq K$. Since edges outside K appear with probability $\frac{1}{2}$, expanding $\Gamma_\cap(Q)$ (line 19) with nodes that are connected to all of $\Gamma_\cap(Q)$ assures that nodes outside K are very unlikely to be added, on the other hand all the nodes of K will be added and so we expect that the set obtained from this process will be with high probability K.

Algorithm PICK (G, α)

denote $s = s(n) = n^{1-c\alpha}$ (c is the constant from Theorem 1.2).
choose uniformly and independently at random with repetitions $\widehat{S} = \{v_1, \ldots, v_s\} \subseteq V$ ($\widehat{S}$ is a multi-set).
for every $j = 1, \ldots, s$ **do**
    set $\beta_j \leftarrow empty - list$
$Q \leftarrow \emptyset$
$i \leftarrow 0$
while $i \leq \binom{s}{m'}$ and $|Q| < B$ **do**
    let $R_i = \{v_{i_1}, \ldots, v_{i_{m'}}\}$ be the i-th subset of $\{v_1, \ldots, v_s\}$ of size m', according to some fixed order.
    let $(t^i_1, \ldots, t^i_{m'}) \leftarrow$ TEST(G, R_i) ▷ t^i_j corresponds to the node v_{i_j}
    if for every $j \in [m']$ $t^i_j \neq 0$ **then**
        $Q \leftarrow Q \cup R_i$
        for every $v_{i_j} \in R_i$ **do**
            add t_j to β_{i_j}
    $i \leftarrow i + 1$
if $|Q| \geq \frac{4}{\alpha}$ **then**
    for every $i = 1, \ldots, s$ **do**
        $t_{v_i} \leftarrow$ the mode (the value that appears most often) of β_i (set 0 if β_i is empty).
    $\Gamma_\cap(Q) \leftarrow \bigcap_{v_i \in Q} \Gamma_{t_{v_i}}(v_i)$ where $\Gamma_{t_{v_i}}(v_i) = \{u \in \Gamma(v) | \tau(u, v_i) = t_{v_i}\}$
    $C \leftarrow \{u \notin \Gamma_\cap(Q) \mid u$ is connected to all the nodes in $\Gamma_\cap(Q)\}$
    return $C \cup \Gamma_\cap(Q)$
else$(|Q| < \frac{4}{\alpha})$
    return $\emptyset$

Note that algorithm PICK(G, α) uses only time-stamps and adjacency relations. Let $(t^S_1, \ldots, t^S_{m'})$ be the output of TEST(G, S), we assume that procedure TEST(G, S) has the following properties:

1. It runs in polynomial time (assuming an edge query is an $O(1)$ operation).
2. It uses only time-stamps and adjacency relations.
3. In every invocation, the algorithm queries only edges adjacent to nodes in S.
4. If $S = \{w_{t_1}, \ldots, w_{t_{m'}}\} \subseteq K^{2/3}$ then with probability at least $1 - \frac{1}{n^r}$ (for an arbitrarily large but fixed $r > 1$) $(t^S_1, \ldots, t^S_{m'}) = (t_1, \ldots, t_{m'})$.
5. Else (i.e. if $S \cap (K \setminus K^{2/3}) \neq \emptyset$) then with probability at least $1 - \frac{1}{n^r}$ (for an arbitrarily large but fixes $r > 1$) $(t^S_1, \ldots, t^S_{m'}) = (0, \ldots, 0)$.

We later show a procedure that meets these conditions (see section 3.3.6).

We now show that with probability at least $1 - \frac{1}{n}$ algorithm PICK(G, α) returns K.

3.3.1 Analysis of steps 7-14 - construction of Q

Claim 3.3. *With probability at least $1-\frac{1}{n^2}$ the set Q computed in step 6 satisfies $Q \subseteq K^{2/3}$ and $|Q| \geq \frac{4}{\alpha}$.*

Proof of Claim 3.3. For every $i \in [\frac{k}{3}+1, \frac{2k}{3}]$ we denote by X_i the random variable indicating if w_i was chosen to be part of $\widehat{S}$ (we note that with some small probability, the same node can be chosen more than once). Let $X = \sum_{w_i \in K^{2/3}} X_i$ the number of nodes from $K^{2/3}$ that were chosen in total.

$$\Pr[X_i = 0] = (1-\frac{1}{n})^s = (1-\frac{1}{n})^{n^{1-c\alpha}} \leq 1 - \frac{1}{n^{(1-c)\alpha}}.$$

So

$$\mathrm{E}[X] = \frac{k}{3}\Pr[X_i = 1] \geq \frac{n^{c\alpha}}{3}.$$

Taking now $\delta = \frac{1}{2}$ and using Chernoff bound we get:

$$\Pr[X < \frac{4}{\alpha}) \leq \Pr[X < \frac{1}{2}\mathrm{E}[X]] \leq \exp(-\frac{n^{c\alpha}}{24}) = \exp(-\Theta(n^{c\alpha})).$$

We get that this probability is $\leq \frac{1}{n^r}$ for an arbitrary fixed $r = \Theta(1)$, in particular for $r = 2$. So with probability at least $1-\frac{1}{n^2}$, in step 7 we check at least one set R_i of size $\frac{4}{\alpha}$ of nodes from $K^{2/3}$. From property 4 of the procedure $\textsc{Test}(G, S)$, we know that with probability at least $1-\frac{1}{n^3}$, for all the nodes in R_i the procedure returns the correct time-stamp in which they "act" and for all other nodes the procedure returns 0. So with probability at least $\geq 1-\frac{1}{n^2}$, the set Q computed in step 7 satisfies $Q \subseteq K^{2/3}$ and $|Q| \geq \frac{4}{\alpha}$, moreover, $Q \subseteq (\cup_{j=1}^{s}\{v_j\}) \cap K^{2/3}$. □

We note that $|Q| < \frac{5}{\alpha}$, since in every iteration of the loop at step 6 at most $m' < \frac{1}{\alpha}$ nodes are added to Q and at the beginning of every iteration $|Q| < \frac{4}{\alpha}$.

3.3.2 Analysis of steps 16-18 - construction of $\Gamma_\cap$

We note that the edges and their labels were used by algorithm Test in order to build the set Q and so we cannot assume that the labels of the edges that are adjacent to $\Gamma_\cap(Q)$ as used in step 18 are distributed according to the distribution induced by the model. To overcome this problem we look at an "idealized" construction of the set Q and note that with high probability the two constructions coincide.

Let $R \subseteq V$ be a multiset of size s that is chosen uniformly at random, possibly with repetitions, and let $R' = R \cap K^{2/3}$. Let $R'_1, \ldots R'_l$ be all the subsets of R' of size m' ordered according to the fixed order (e.g. lexicographic) used by the algorithm at step 8. Let d be the smallest number such that $|\bigcup_{i=1}^{d} R'_i| \geq \frac{4}{\alpha}$ where the union is without repetitions, and let $\widehat{Q} = \bigcup_{i=1}^{d} R'_i = \{w_{t_1}, \ldots, w_{t_q}\}$. Note that $\frac{4}{\alpha} \leq q \leq \frac{5}{\alpha}$. We note that according to the analysis of step 7 and assumptions regarding procedure $\textsc{Test}(G, S)$, with high probability the set Q computed by the algorithm is equal to $\widehat{Q}$. From this point we consider this is indeed the case

($Q = \widehat{Q}$) and analyze $\Gamma_\cap(\widehat{Q})$ instead of $\Gamma_\cap(Q)$ as defined in the algorithm. The key aspect in this analysis is that $\widehat{Q}$ is defined without looking at time-stamps of edges.

For a node $w_l \in K^{3/3}$ and for every $1 \leq j \leq q$ we first note that $(w_l, w_{t_j}) \in E$ (since $w_l, w_{t_j} \in K$) and

$$\Pr[w_l \in \Gamma_{t_j}(w_{t_j})] = (1-p)^{t_j - 1} \geq 0.6.$$

The inequality is due to $t_j \in [\frac{k}{3} + 1, \frac{2k}{3}]$. Denote by Z_l the random variable indicating if $w_l \in \Gamma_\cap(\widehat{Q})$ and by $Z = \sum_{w_l \in K^{3/3}} Z_l$ the number of nodes from $K^{3/3}$ that are in $\Gamma_\cap(\widehat{Q})$. We get:

$$\Pr[Z_l = 1] = \Pr[\forall j \in [q],\ \tau(w_l, w_{t_j}) = t_j] = \prod_{j=1}^{\frac{4}{\alpha}} (1-p)^{t_j - 1} \geq 0.6^q.$$

$$\mathrm{E}[Z] \geq 0.6^q \frac{k}{3}.$$

We note that $Z_{\frac{2k}{3}+1}, \ldots, Z_k$ are independent, so we can use Hoeffding's inequality. Taking $t = \sqrt{\frac{kr \ln n}{6}}$ for $r = \Theta(1)$:

$$\Pr[|Z - \mathrm{E}([Z]| \geq t] \leq 2\exp(-\frac{6t^2}{k}) \leq \frac{1}{n^r}$$

We note that since $r = \Theta(1)$,

$$\mathrm{E}[Z] - t \geq 0.6^q \frac{k}{3} - t \geq \frac{k}{2^{\frac{16}{\alpha}}}.$$

Taking $r = 2$ we get that with probability at least $1 - \frac{1}{n^2}$, $Z \geq \frac{k}{2^{\frac{4}{\alpha}}}$, i.e. the set $\Gamma_\cap(Q)$ in the algorithm, contains a constant fraction of K.

On the other hand, for a node $u \notin K$ we have:

$$\Pr[u \in \Gamma_\cap] = \prod_{j=1}^{q} \Pr[\tau(u, w_{t_j}) = t_j] = \prod_{j=1}^{q} (1-p)^{t_j - 1} p \leq p^q \leq p^{\frac{4}{\alpha}} \leq (\frac{\ln 2}{n^\alpha})^{\frac{4}{\alpha}}.$$

Using a union bound over all $u \notin K$, we get that (for large enough values of n)

$$\Pr[\exists u \notin K \text{ such that } u \in \Gamma_\cap] \leq n(\frac{\ln 2}{n^\alpha})^{\frac{4}{\alpha}} = \frac{(\ln 2)^{\frac{4}{\alpha}}}{n^3} \leq \frac{1}{n^2}.$$

3.3.3 Analysis of steps 19-20- constriction of C

We note that the edges and their labels were used by algorithm TEST in order to build the set Q, and so the set $\Gamma_\cap(Q)$ is not random and we can not assume that the edges adjacent to $\Gamma_\cap(Q)$ are random and that each edge appear with probability exactly $\frac{1}{2}$.
To overcome this problem we look at the set $Q' = \widehat{S} \cap K^{2/3}$ and we denote by $w_{r_1}, \ldots, w_{r_{q'}}$

the nodes of Q' and assume (for the purpose on this proof) that $r_1, \ldots, r_{q'}$ are known. We note that since $\widehat{S}$ is chosen uniformly at random, the set Q' is a set that is chosen uniformly at random from $K^{2/3}$ and we did not look at it's outgoing edges.

For $R \subseteq Q$ such that $\frac{4}{\alpha} \leq |R| \leq \frac{5}{\alpha}$ we look at $\Gamma_\cap(R)$. From the analysis of steps 16-18 we know that for a fixed R with high probability $\Gamma_\cap(R) \subseteq K$ and $|\Gamma_\cap(R)| \geq \frac{k}{2^{\frac{16}{\alpha}}}$. We note that since $\widehat{S}$ is chosen uniformly at random, the sets Q', R are chosen uniformly at random from $K^{2/3}$ and so $\Gamma_\cap(R)$ is a set in K which is independent of edges outside of K. The probability for a node $u \notin K$ to connect to all of $\Gamma_\cap(R)$ is at most $(\frac{1}{2})^{|\Gamma_\cap(R)|} \leq (\frac{1}{2})^{\frac{k}{2^{\frac{16}{\alpha}}}} \leq \frac{1}{n^3}$, so with probability at least $1 - \frac{1}{n^2}$ no node outside K is connected to all of $\Gamma_\cap(R)$. On the other hand every $u \in K$ is connected to all the nodes in $\Gamma'_\cap$.
Since $\textsc{Test}(G, Q)$ returns the right answer with high probability, we get that with high probability $Q \subseteq Q'$ i.e. $Q = R$ for some $R \subseteq Q'$ and $\Gamma_\cap(Q) = \Gamma_\cap(R)$, so from the above analysis, we get that exactly the nodes of K will be added to C.

3.3.4 Success probability

For the algorithm to fail at least one of the following must happen:

- Less than $\frac{4}{\alpha}$ distinct nodes from $K^{2/3}$ are chosen to $\widehat{S}$.
- Procedure $\textsc{Test}(G, Q)$ returns a wrong value (in any of its $\binom{s}{m'}$ invocations).
- The set $\Gamma_\cap(Q)$ is not a constant fraction of K (i.e. $|\Gamma_\cap(\widetilde{Q})| < \frac{k}{2^{\frac{16}{\alpha}}}$).
- There exists a node $v \notin K$ such that $v \in \Gamma_\cap(Q)$.
- $Q \nsubseteq Q'$.
- There exists a node $v \notin K$ such that $v \in C$.

Each of these events happens with probability at most $\frac{1}{n^2}$, so overall the algorithm succeeds with probability at least $1 - \frac{1}{n}$.

3.3.5 Number of queries and running time

Algorithm $\textsc{Test}$ uses queries exactly to all the neighbors of $\widehat{S}$. Each node has at most $n - 1$ neighbors, there are s nodes so in total $s(n-1) = \Theta(n^{2-c\alpha})$ queries are needed. We note that in order to not query edges more then once (and thus using more queries), one can implement a global data structure that saves all answers from past queries (this data structure will be used by $\textsc{Pick}$).

Denoting $T_{\textsc{Test}}$ the runtime of $\textsc{Test}(G, Q)$, it is easy to see that the runtime of $\textsc{Pick}(G, \alpha)$ is $O(s^{m'} T_{\textsc{Test}})$, we assume that $T_{\textsc{Test}}$ is polynomial so for constant α the runtime is polynomial. □

3.3.6 Procedure Test(G, Q)

We describe now the procedure Test(G, Q). The procedure can distinguish whether the given set of nodes Q is in K or is outside (or only partly in) K by looking at the frequencies that some time-stamps appear in edges adjacent to the nodes of Q. The procedure gets G and a set $Q = \{v_1, \ldots, v_M\} \subseteq V$ of size M (to be chosen later by Claim 3.4) and returns a vector of M values $(t_1^Q, \ldots, t_M^Q)$. The procedure Test uses a function d that is defined on sets of nodes of size M, as described below in Claim 3.4, and is computable in polynomial time. The procedure counts for every set $S = \{t_1, \ldots, t_M\}$ how many nodes $u \in V$ are there such that for every $i \in [M]$ they are connected to v_i by an edge with time-stamp t_i (steps 2-3). Intuitively if $Q \subseteq K$ and $\{t_1, \ldots, t_M\}$ are exactly the nodes' names (i.e. $v_i = w_{t_i}$) then there are many such nodes u (nodes in the clique) and in other cases there will be only few such nodes. The parameter d represents a threshold of the described number, i.e. pairs of sets Q, S for which the number of such nodes u is larger than d is likely to be such that $Q \subseteq K$ and that the names of the nodes are according to S and otherwise the names are not according to S or the set is not a subset of K.

Algorithm Test (G, Q)

1: query $\tau(v_l, u)$ for every $v_l \in Q$ and $u \in V \setminus Q$ (if u, v are not connected then $\tau(u, v) = 0$)
2: **for** every set $S = \{t_1, \ldots, t_M\} \subseteq [k]$ **do**
3: let $b_Q(S) = |\{u \in V \setminus Q \mid \forall 1 \leq l \leq M,\ \tau(v_l, u) = t_l\}|$
4: let $S_Q = \arg\max_{S \subseteq [k]} \{b_Q(S)\}$
5: **if** $b_Q(S_Q) \geq d(S_Q)$ **then**
6: **return** $S_Q = (t_1, \ldots, t_M)$
7: **else**
8: **return** $(0, \ldots, 0)$

Note that procedure Test(G, S) uses only time-stamps and adjacency relations. We define the following variables for the analysis of the procedure. For nodes $u, v \in V$ let $Y_v^t(u)$ be a random variable indicating if $\tau(u, v) = t$. For a set of nodes $Q = \{v_1, \ldots, v_M\}$ (all distinct), a corresponding set of time-stamps $S = \{t_1, \ldots, t_M\}$ (all distinct) and a node $u \notin Q$, we define the random variable $Y_Q^S(u)$ indicating if $\tau(u, v_i) = t_i$ for every $1 \leq i \leq M$ (if the edge does not exists then we assume $\tau(u, v_i) = 0$). Observe that $Y_Q^S(u) = \prod_{i=1}^M Y_{v_i}^{t_i}(u)$ and from the independence of the edge labels,

$$\Pr[Y_Q^S(u) = 1] = \prod_{i=1}^{M} \Pr[Y_{v_i}^{t_i}(u) = 1].$$

This equality is correct for both cases $u \in K$ and $u \notin K$, each case has to be verified (via an easy calculation) separately. The probability distributions for the variables $Y_v^t(u)$ and $Y_Q^S(u)$ appear in Appendix A.

We now look at the random variable $Y_Q^S = \sum_{u \in V} Y_Q^S(u)$. We define 4 basic types of pairs $\langle Q, S \rangle$ where $S = \{t_1, \ldots, t_M\}$:

1. $Q = \{w_{t_1}, \ldots, w_{t_M}\} \subseteq K$.
2. $Q = \{v_1, \ldots, v_M\} \subseteq V \setminus K$.
3. $Q = \{w_{i_1}, \ldots, w_{i_M}\} \subseteq K$ such that $\forall l \in [M]$ $t_l < i_l$
4. $Q = \{w_{i_1}, \ldots, w_{i_M}\} \subseteq K$ such that $\forall l \in [M]$ $t_l > i_l$

We note that if $u \notin K$ then $Y_Q^S(u)$ distribution is the same for all $\langle Q, S\rangle$ are of different type. The effect of the different types appear only for $u \in K$.

From these basic types, we can build composite types of pairs $\langle Q, S\rangle$. Each pair $\langle Q, S\rangle$ can be broken into disjoint pairs $\langle Q'_i, S'_i\rangle$ of type i for $1 \leq i \leq 4$ such that $|Q'_i| = |S'_i| = m_i$. For the sake of analysis, we assume WLOG that $Q = \{v_1, \ldots, v_M\}$ such that $v_1, \ldots, v_{m_1} \in Q'_1$, $v_{m_1+1}, \ldots, v_{m_1+m_2} \in Q'_2$, $v_{m_1+m_2+1}, \ldots, v_{m_1+m_2+m_3} \in Q'_3$, $v_{m_1+m_2+m_3+1}, \ldots, v_m \in Q'_4$. This random variable can be written as:

$$Y_Q^S = \sum_{u \in V} Y_Q^S(u) = \sum_{u \in V} \prod_{i=1}^{4} Y_{Q'_i}^{S'_i}(u).$$

We note that if $m_4 > 0$ the random variable "behaves" exactly as if $m_4 = M$ (since it can not connect to nodes in K with the right time-stamp), i.e. $Y_Q^S(u) = 0$. So from now on we identify sets such that $m_4 > 0$ with sets of the form $m_4 = M$, and assume that composite sets are broken into only the first three types.

Claim 3.4. *Let $S = \{t_1, \ldots, t_M\} \subseteq \{\frac{k}{3} + 1, \ldots, \frac{2k}{3}\}$ and M be the largest integer such that $M \leq \frac{1}{\alpha} - 1$, there exists a computable $d = d(S) =$ such that with probability at least $1 - \frac{1}{n^r}$ for an arbitrarily large but fixed r:*

- $Y^S_{\{w_{t_1}, \ldots, w_{t_M}\}} > d$ *and*
- $Y_Q^S < d$ *for every $Q \subseteq V$, $|Q| = M$ such that $Q \neq \{w_{t_1}, \ldots, w_{t_M}\}$.*

Proof of Claim 3.4. Fix some $S = \{t_1, \ldots, t_M\} \subseteq \{\frac{k}{3}+1, \ldots, \frac{2k}{3}\}$ and let $Q_1 = \{w_{t_1}, \ldots, w_{t_M}\}$ and Q_i, $|Q_i| = M$ for $2 \leq i \leq 4$ be any fixed set such that $\langle Q_i, S\rangle$ is a pair of type i. Let Q_5 be a set such that $\langle Q_5, S\rangle$ is of a composite type (at least two of m_1, m_2, m_3 are greater than 0). We denote by $\mu_i = E(Y_{Q_i}^S)$ for $i \in [5]$.

Lemma 3.5. $\mu_4, \mu_3, \mu_2 \leq \mu_5 \leq \mu_1$.

We prove Lemma 3.5 in Appendix A.

Since Lemma 3.5 is true for any 5 sets (of the proper types) it's also true for

$$Q'_5 = Q'_5(S) = \arg \max_{Q_5 \text{ is composite with respect to } S} \{E(Y_{Q_5}^S)\}$$

Let $\mu'_5 = E(Y_{Q'_5}^S)$. We denote by $\delta_i = \mu_1 - \mu_i$ for $2 \leq i \leq 5$ and $\delta'_5 = \mu_1 - \mu'_i$, and let $d_i = \frac{\mu_1+\mu_i}{2} = \mu_i + \frac{\delta_i}{2}$ and $d'_5 = \frac{\mu_1+\mu'_5}{2} = \mu_5 + \frac{\delta'_5}{2}$. From Lemma 3.5 we get that $d_2, d_3, d_4 \leq d_5 \leq d'_5$. We will see:

- $\Pr[Y^S_{Q_1} \geq d'_5] \geq 1 - \frac{1}{n^{r'}}$ (for an arbitrary fixed $r' = \Theta(1)$).
- For every $Q \neq Q_1$, $\Pr[Y^S_Q \leq d'_5] \geq 1 - \frac{1}{n^{r'}}$ (for an arbitrary fixed $r' = \Theta(1)$).

Since there are at most n^M sets $Q \neq Q_1$ and $M = \Theta(\frac{1}{\alpha})$, taking $r' = M + r$ we get that with probability at least $1 - \frac{1}{r}$ for all $Q \neq Q_1$ it holds that $Y^S_Q \leq d'_5$ and so d'_5 has the separating property needed, i.e. with high probability:

- $Y^S_{Q_1} > d'_5$
- $Y^S_Q < d'_5$ for every $Q \subseteq V$, $|Q| = M$ such that $Q \neq Q_1$.

We show that d'_5 is computable in polynomial time. For a set $S \subseteq \{\frac{k}{3} + 1, \ldots, \frac{2k}{3}\}$ we denote by $\rho = \rho(S) = \max\{t_i \in S\}$ and get:

$$\mu_1 = E(Y^S_{Q_1}) = (n-k)(1-p)^{\sum_{l=1}^{<} t_i - M} p^M + (k-\rho)(1-p)^{\sum_{l=1}^{<} t_i - M},$$
$$\mu_5 = E(Y^S_{Q_5}) = (n-k)(1-p)^{\sum_{l=1}^{M} t_l - M} p^M + (k - \rho - 1 + p^{-I_{\{m_3>0\}}})(1-p)^{\sum_{l=1}^{M} t_l - M} p^{m_2+m_3}.$$

Where $I_{\{m_3>0\}} = 1$ if $m_3 > 0$ and $I_{\{m_3>0\}} = 0$ otherwise (a detailed explanation of the computation of μ_5 appears in Appendix A). In order to maximize μ_5 we need to minimize the power in the expression $p^{m_2+m_3}$, we know that it must be at least 1 (otherwise the set will be of type 1 and not composite), so we take $m_2 + m_3 = 1$, we note that the option of $m_2 = 0, m_3 = 1$ is better since in this case we get

$$k - \rho - 1 + p^{-I_{m_3>0}} = k - \rho - 1 + \frac{1}{p}.$$

So we see that:

$$d'_5 = \frac{2(n-k)(1-p)^{\sum_{i=1}^{M} t_i - m} p^M + (k-\rho)(1-p)^{\sum_{i=1}^{M} t_i - M} + (k - \rho - 1 + \frac{1}{p})(1-p)^{\sum_{i=1}^{M} t_i - m} p}{2}$$

This is indeed computable in polynomial time. We turn to show that with high probability $Y^S_{Q_1} > d'_5$ and $Y^S_Q < d'_5$ for every $Q \subseteq V$, $|Q| = M$ such that $Q \neq Q_1$. Taking $\epsilon_1 = \frac{\delta_5}{2\mu_1} = \frac{\mu_1 - \mu_5}{2\mu_1} < 1$, and using Chernoff bound:

$$\Pr[Y^S_{Q_1} \leq d'_5] \leq \Pr[Y^S_{Q_1} \leq (1-\epsilon_1)\mu_1] \leq \exp(-\frac{\epsilon_1^2 \mu_1}{2}) = \exp(-\frac{\delta_5^2}{8\mu_1})$$
$$\leq \exp(-\frac{k^2 (\frac{1}{2})^{\frac{2M}{3}}}{256 n p^M}) = \exp(-\Theta(n^{(M+2)\alpha - 1})).$$

The last inequality is due to:

- $\mu_1 \leq (k + np^M)(1-p)^{\sum_{l=1}^{M} t_i - M}$ and since $M < \frac{1}{\alpha} - 1$ we have $k < np^M$ so $\mu_1 \leq 2np^M$.
- $\delta_5 \geq \frac{k}{4}(1-p)^{\sum_{l=1}^{M} t_i - M} \geq \frac{k}{4}(\frac{1}{2})^{\frac{2M}{3}}$.

Since $M > \frac{1}{\alpha} - 2$ is a constant, for every $\alpha < \frac{1}{2}$ the above probability is $\exp(-\Theta(n^{\beta}))$ for some constant β so, this probability is $\leq \frac{1}{n^{r'}}$ for an arbitrary fixed $r' = \Theta(1)$.

Taking $\epsilon_5 = \frac{\delta_5}{2\mu_5}$ and using Chernoff bound:

$$\Pr[Y^S_{Q_5} \geq d'_5] \leq \Pr[Y^S_{Q_5} \geq (1+\epsilon_5)\mu_5] \leq \exp(-\frac{\epsilon_5^2\mu_5}{3}) = \exp(-\frac{\delta_5^2}{12\mu_5})$$
$$\leq \exp(-\frac{k^2(\frac{1}{2})^{\frac{2M}{3}}}{384np^M}) = \exp(-\Theta(n^{(M+2)\alpha-1})).$$

The last inequality is due to:

- $\mu_5 \leq \mu_1 \leq 2np^M$ since $M \leq \frac{1}{\alpha} - 1$.
- $\delta_5 \geq \frac{k}{4}(1-p)^{\sum_{l=1}^{M} t_i - M} \geq \frac{k}{4}(\frac{1}{2})^{\frac{2M}{3}}$.

As before, this probability is $\leq \frac{1}{n^{r'}}$ for an arbitrary fixed $r' = \Theta(1)$.

We note that Q_2, Q_3, Q_4 are stochastically dominated by Q_5 hence the same tail bound applies for $\Pr[Y^S_{Q_i} \geq d_5]$ for every $2 \leq i \leq 4$.

□

Claim 3.6. *Denoting $(t^Q_1, \ldots, t^Q_M) \leftarrow \textsc{Test}(G, Q)$, procedure $\textsc{Test}(G, Q)$ satisfies the following properties:*

1. *It runs in polynomial time (assuming an edge query is an $O(1)$ operation).*
2. *It uses only time-stamps and adjacency relations.*
3. *In every invocation, the algorithm queries only edges adjacent to nodes in S.*
4. *If $Q = \{w_{t_1}, \ldots, w_{t_M}\} \subseteq K^{2/3}$ then with probability at least $1 - \frac{1}{n^r}$ (for an arbitrarily large but fixed $r > 1$) for every $l \in [M]$ it holds that $t^Q_l = t_l$.*
5. *else, i.e. if $Q \cap K \setminus K^{2/3} \neq \emptyset$, with probability at least $1 - \frac{1}{n^r}$ (for an arbitrarily large but fixed $r > 1$) for every $l \in [M]$ it holds that $t^Q_l = 0$.*

Proof of Claim 3.6. Let us analyze the running time: In step 1, there are at most $|Q|n = Mn = O(n)$ queries, each takes $O(1)$ time. In step 2, we compute $\binom{k}{M} = O(n^{\alpha M})$ sets $b_Q(S)$, each set is compute in $O(n)$ time. From Theorem 3.4, we know that for every $S \subseteq [k]$ the value $d(S_Q)$ is computable in polynomial time. So the entire algorithm takes polynomial time.

Properties 2, 3 are clear from the algorithm itself. Properties 4, 5 follow directly from Theorem 3.4. □

3.4 Lower bound

Here we refer to our model (with time-stamps) of the hidden clique problem as the temporal model and to the model without time-stamp as the classical model. We start by proving a lower bound for deterministic algorithms we then extend this to randomized algorithm

Theorem 3.7. *Suppose there exists a deterministic polynomial time algorithm A that gets G as input and uses $s = O(\log n)$ adaptive edge queries to solve the hidden clique problem in the edge queries model with high probability. Then there exists a probabilistic polynomial time algorithm B that solves the hidden clique problem in the classic model (i.e. with no edge queries) with at least the same probability.*

We first clarify the meaning of an algorithm that uses "deterministic adaptive edge queries". By this we mean that the choice of the i-th edge to be queried may depend on the answers of the queries done before, but it is deterministic, i.e. if we execute the algorithm more than once and get the same answers to the first $i-1$ queries, the i-th edge to be queried will be the same in all the executions (with probability 1). So algorithm A has WLOG the following form:

Algorithm $A(G)$

1: $PrevQueries \leftarrow empty - stack$
2: **for** $i = 1, \ldots, m$ **do**
3: $\quad e_i \leftarrow A^*(G, PrevQueies, i)$
4: $\quad$ query edge e_i and denote by t_i the answer
5: $\quad push(PrevQueries, (e_i, t_i))$
6: **return** $A'(G, PrevQueries)$

Here A^* and A' are deterministic polynomial time that do not make any queries, and with high probability (that is taken only over the input graph and its time-stamps) $A'(G, PrevQueries) = K$ when $PrevQueries$ is generated properly. We assume WLOG that no edge is queried more than one.

Proof of Theorem 3.7. We denote by Q the distribution of time-stamps for an edge that at least one of its end nodes is outside K, and by $\widehat{Q}_{t'}$ the distribution of time-stamps for an edge that both of its end nodes are in K and the endpoint that is added to K first, is added at step t'. We note that both distributions are samplable in polynomial time. From here on, we refer to the time in which a node in the clique acts as its name, nodes that are outside the clique are all named 0.

We start by explaining the general scheme of algorithm B. The algorithm traverse through all root-leaf paths using a DFS, in a quadtree (a tree in which each node has exactly four children) with $m+1$ levels (starting at level 0). At level $i > 0$ the algorithm executes A^* with input the graph G, the edges queried so far with the time-stamps generated for them (in the current path from the root) and i - the number of the step. Let $e = (u, v)$ be the edge returned by A^* at a certain node in level i. Since we don't know if u, v are both inside

the clique, both outside it or one is in the clique and the other is outside, we check all four options.Each child in the tree represent one of these four options:

1. *Both nodes are inside the clique.* If any of the nodes was already a part of a queried edge we need to be consistent with the name given to it before. In particular, if any of the nodes was named before we must keep that name. On the other hand, if a node was decided previously to be outside the clique we can't be consistent and the search in that subtree is stopped. If a node has not been decided yet, assign it a random name that was not used for nodes in this path, denote the names by $Name(u), Name(v)$. This is done in the procedure $GenerateNamesClique(G, C, u, v)$ described below.

2. *Both nodes are outside the clique.* First make sure that both nodes were not decided before to be in the clique. If one of the nodes was decided to be in the clique we can't be consistent and the search in that subtree is stopped. Otherwise, give both the name 0. This is done in the procedure $GenerateNamesNotClique(G, C, u, v)$ described below.

3. *One node is inside the clique and the other is outside the clique.* Generally there are two sub-cases, the first is if u is decided to be in the clique and v is decided to be outside the clique, in the second the decisions are swapped. In the algorithm the sub-cases are treated separately. Here we describe only the first, the second is treated analogously. If u was decided before to be outside the clique or v was decided to be in the clique we can't be consistent and the search in that subtree is stopped. If u was already assigned a name keep this assignment, otherwise assign it a random name that was not used for nodes in this path. Give v the name 0. This is done in the procedure $GenerateNamesMixed(G, C, u, v)$ described below.

After deciding which option and generating the names (assuming we can be consistent) we can create a time-stamp for the edge according to its type (outside the clique according to the distribution Q and inside the clique according to $\widehat{Q}_{\min\{Name(u),Name(v)\}}$). This generated data is then added to $PrevQueries$ (with the edge that it labels), to be used in the next iteration of the algorithm. When reaching a leaf we use all data generated so far as input to A' and check if the output is indeed the clique, if so we return it, otherwise we continue to the next root-leaf path. If after scanning all the root-leaf paths in the tree the clique was not found we return "failure" and stop.

We turn now to a formal description of algorithm B that uses three procedures (that were described before):

Algorithm B

let $\widehat{T}$ be full quadtree (i.e. each node of the tree has exactly 4 children) with $m+1$ levels (starting at level 0).
initialize an empty stack $PrevQueries$.
initialize an array $Name$ indexed by all the nodes of G to have -1 in each cell.
for every node y in T according to a DFS order **do**
    if y is at level $i > 0$ of the tree and is entered for the first time **then**
        $e(y) \leftarrow A^*(G, PrevQueries, i)$ and denote $e(y) = (u^0(y), u^1(y))$
        if y is the first child of y' (we assume that $u^0(y), u^1(y)$ are in the clique) **then**
            GenerateNamesClique$(G, Name, u^0(y), u^1(y))$
            if the procedure returned "failure" **then**
                stop scanning the subtree of y
            else
                choose $t'(e(y))$ according to the distribution $\widehat{Q}_{\min\{Name(u^0(y)),Name(u^1(y))\}}$
        else
            if y is the second child of y' (we assume that $u^0(y), u^1(y)$ are not in the clique) **then**
                GenerateNamesNotClique$(G, Name, u^0(y), u^1(y))$
                if the procedure returned "failure" **then**
                    stop scanning the subtree of y
            else if y is the third child of y' (we assume that $u^0(y)$ is in the clique and $u^1(y)$ is not in the clique) **then**
                GenerateNamesMixed$(G, Name, u^0(y), u^1(y))$
                if the procedure returned "failure" **then**
                    stop scanning the subtree of y
            else if If y is the fourth child of y' (we assume that $u^0(y)$ is not in the clique and $u^1(y)$ is in the clique) **then**
                GenerateNamesMixed$(G, Name, u^1(y), u^0(y))$
                if the procedure returned "failure" **then**
                    stop scanning the subtree of y
            $push(PrevQueries, (e(y), t'(e(y))))$
            if y is a leaf (i.e. i=m) **then**
                if $M = A'(G, PrevQueries)$ is a clique of size n^α **then**
                    return M
    else if y is left for the last time (i.e. the search in the subtree of y is finished) **then**
        $pop(PrevQueries)$
        for $r \in \{0, 1\}$ **do**
            if $u^r(y)$ dose not appear in any edge that is in $PrevQueries$ **then**
                Set $Name(u^r(y)) = -1$
return failure

We describe the three procedures:

Algorithm GenerateNamesClique(G, C, u, v)

if $C(u) = 0$ or $C(v) = 0$ **then**
  return failure
for $\sigma \in \{u, v\}$ **do**
  if $C(\sigma) = -1$ **then**
    set $C(\sigma)$ to a uniformly random value from $[n^\alpha]$ that does not appear in C

Algorithm GenerateNamesNotClique(G, C, u, v)

if $C(u) > 0$ or $C(v) > 0$ **then**
  return failure
for $\sigma \in \{u, v\}$ **do**
  set $C(\sigma) = 0$

Algorithm GenerateNamesMixed(G, C, u, v)

if $C(u) = 0$ or $C(v) > 0$ **then**
  return failure
if $C(u) = -1$ **then**
  set $C(u)$ to a uniformly random value from $[n^\alpha]$ that does not appear in C
set $C(v) = 0$

We first note that algorithm B runs in polynomial time. The tree $\widehat{T}$ has $O(4^m) = O(2^{O(\log n)}) = n^{O(1)}$ root-leaf paths, for each such path a polynomial amount of work is done.

Notice that the success probability of algorithm A is taken over the input graph (i.e. which random edges appear and which set is chosen to be the clique), the distribution of time-stamps of edges (given the graph). The success probability of algorithm B is taken over the input graph and the randomization used in it (B's coin-tosses).

The names of nodes and the time-stamps that are given with the graph are called in this proof the real names and time-stamps (respectively).

We turn to prove the correctness of this algorithm. We do so by induction over the number of iteration (query). We show that there exists a path in which the generated names and time-stamps are distributed exactly as the real names and time-stamps and so when computing A' at the leaf at the end of this path, the success probability is the same as the success probability of algorithm A.

Base case: For $i = 1$, the edge must be of one of the types and so going to the corresponding child in the tree is a correct path. The time-stamp distributions:

- If the edge is outside the clique, in the corresponding path in the tree we generate the time-stamp according to Q which is the right distribution.

- If the edge is inside the clique then since each node in the clique is as likely as any other node to have any name and since we generate the names at random and without repetitions we generate the names according to the real distributions. Since the names are generated correctly, we use a correct distribution to generate the time-stamp.

Inductive step: Assume that we have a path of length $i < m$ in which the choices that were made are correct (for all nodes that were involved in a query we guessed correctly if they are in the clique or not) and the names and time-stamps were, so far, generated according to the real distribution, we show that we can continue this path.

The new edge that is queried must be again one of the 4 types so going to the corresponding child in the tree will yield a correct path. We note that for every two paths that have a common segment, the algorithm uses the same names and time-stamps all along it, this way we make sure that all four possibilities for a specific edge are indeed checked. Looking at the names given to the nodes of the edge, since we assume that we are in the correct child, names from the correct range are given (i.e. 0 to nodes outside the clique and a new name from $[n^{\alpha}]$ to nodes inside the clique). As to the distribution of names of nodes in the clique, since the names are chosen uniformly at random each assignment for all the nodes involved is equally likely to be chosen by the algorithm as the real distribution. Since the names of the nodes are distributed according to the real distributions of name, the distributions according to which the time-stamps are generated are also compatible with the real distributions.

We notice that the algorithm keeps consistency with past choices of names, so if a node is involves in more than one queried edge in the distributions used to generate the time-stamps for all these edges we use the same name for that node, as should be. So we get that the time-stamps generated are distributed as in the real time-stamps.

From the above, there exists a root-leaf path such that all time-stamps needed along it are generated in a distribution identical to the real distribution of time-stamps and so when executing A' with the data generated along this path the success probability is as in the case that A' is executed on real data.

We get that the total success probability of algorithm B is at least as high as the success probability of algorithm A, since we have at least one path in which the probability is the same and other paths can not decrease the success probability because each possible answer is checked (and if it is not the clique we continue to the next path). □

3.4.1 Extension to randomized algorithms

We now extend this proof for the randomized algorithms

Theorem 1.3. *Suppose there exists a randomized polynomial time algorithm A_{rand} that gets G as input and uses $s = O(\log n)$ adaptive edge queries to solve the hidden clique problem in the edge queries model with high probability. Then there exists a probabilistic polynomial time algorithm B_{rand} that solves the hidden clique problem in the classic model (i.e. with no edge queries) with at least the same probability.*

We clarify the meaning of "randomized algorithm that uses adaptive edge queries". By this we mean that the choice of the i-th edge to be queried may depend on the answers of

the queries done before as well as on random coin tosses it makes. So algorithm A_{rand} has WLOG the following form:

Algorithm A_{rand}

$PrevQueries \leftarrow empty - stack$
for $i = 1, dots, m$ **do**
    $e_i \leftarrow A^*_{rand}(G, PrevQueries, i)$
    query edge e_i and denote by t_i the answer
    $push(PrevQueries, (e_i, t_i))$
return $A'_{rand}(G, PrevQueries)$

Here A^*_{rand} and A'_{rand} are probabilistic polynomial time algorithm that do not make any queries, and with high probability (taken only over the input graph, its time-stamps and the random coins used) $A'_{rand}(G, PrevQueries, Randomness) = K$ when $PrevQueries$ is generated properly.

Proof sketch of Theorem 1.3. Algorithm B_{rand} is exactly the same as in the deterministic case and the correctness is as before, we note that if we guess all the types of the edges correctly then all the data used in all the execution is distributed at the real graphs and the randomness used by the algorithm is also the same. So the total success probability of algorithm B_{rand} is, as before, at least as high as the success probability of algorithm A_{rand}. □

4 Hidden clique problem in snapshots model

In this section, the graph G, the clique $K = \{w_1, \ldots, w_k\}$ and the time-stamps of the edges are generated and distributed exactly as in section 3. Specifically the graph is constructed over k discrete steps, at each step $1 \leq t \leq k$ the node w_i of the clique connects to all other nodes of the clique (that it is not already connected to) and any pair of nodes $u, v \in v$ that are not connected already are connected with probability $p = 1 - (\frac{1}{2})^{\frac{1}{k}} = 1 - \Theta(\frac{1}{k})$ independently of all other events. We denote by $\tau(u, v)$ the step in which the nodes u, v were connected ($\tau(u, v) = 0$ if $(u, v) \notin E$) and by E_t the set of edges that were added to G at step t.

The difference from section 3 is that now we use the snapshot query method for accessing the temporal information, i.e. a single query provides the set E_t of all edges in G that were added in step $1 \leq t \leq k$.

Recall that by Lemma 3.2, the resulting graph without the edge labels is an instance of the hidden clique problem with parameters $(G, \frac{1}{2}, n^\alpha)$. For constant $\alpha < \frac{1}{2}$ there are no known polynomial algorithm to this problem. Since there are efficient algorithms that find K with high probability (without the temporal information) when $\alpha \geq \frac{1}{2}$, we focus on the case that $\alpha < \frac{1}{2}$.

In section 4.1 we show that even one snapshot may provide enough information for finding K, under the conditions that the clique is relatively large (i.e. $k = \widetilde{\Omega}(n^{\frac{1}{3}})$) and that the snapshot given is not of a too close to the last step k. In section 4.2 we show that given a logarithmic number of snapshot any polynomial size K can be found with high probability.

4.1 Algorithm that uses one query

Theorem 1.4. *Assume $k \geq cn^{\frac{1}{3}} \ln n$ for large enough constant $c > 0$ and fix a constant $0 < \beta < 1$. Then for any $t \leq \beta k$, given G and E_t one can find K in polynomial time with high probability.*

Proof of Theorem 1.4. We present algorithm OnePoint that gets G and E_t for some $t \leq \beta k$ and returns K with high probability. The algorithm iterates over all nodes u, for each node we look at H_u the subgraph of G (with edges from all steps) induced by the neighbors of u in E_t and we look for a hidden clique in it. If a clique C_u is found we try to expand it by adding each node from $V \setminus V(H_u)$ that is connected to all of C_u. If during this process a clique of size k is found we return it and stop.

The idea is that when $u = w_t$, H_u will have a large clique in it that with high probability will be a constant fraction of K, and so expanding it as suggested will yield K.

Algorithm OnePoint (G, E_t)

1: **for** every $u \in V$ **do**
2: let $\Gamma_t(u)$ be the neighbors of u in E_t and let $H_u = G[\Gamma_t(u)]$ the subgraph of G induced by $\Gamma_t(u)$
3: find a hidden clique in H_u using any of the known algorithms (for $\alpha \geq \frac{1}{2}$) and let C_u be it's output
4: $K_u \leftarrow C_u$
5: **for** every $v \in V \setminus \Gamma_t(u)$ **do**
6: **if** v is connected to all of C_u **then**
7: $K_u \leftarrow K_u \cup \{v\}$
8: **if** K_u is a clique of size k **then**
9: **return** K_u
10: **return** $\emptyset$

Note that for every $u \in V$ H_u includes edges from all of E, not only from E_t.
We now focus on the iteration of loop 1 in which $u = w_t$.

Claim 4.1. *$G[\Gamma_t(w_t)]$ is an instance of the hidden clique problem, and with probability at least $1-\frac{1}{n}$, it has parameters $(M, \frac{1}{2}, Q)$, with $Q = \Omega(\sqrt{M})$ and so can be solved in polynomial time with high probability. Moreover, the hidden clique in $G[\Gamma_t(w_t)]$ is $K \cap \Gamma_t(w_t)$ and with high probability its size is $\Theta(k)$.*

Proof of Claim 4.1. We denote by $\Delta_t(v)$ the number of edges adjacent to v that first appear at time t i.e. $\Delta_t(v) = |\{u \in V | \tau(u,v) = t\}|$. Taking $s = \sqrt{n \ln n}$ and using Hoeffding's inequality we get

$$Pr(\Delta_t(w_t) - E(\Delta_t(w_t)) > s) \leq \exp(-\frac{2s^2}{n-t}) = \exp(-\frac{2n \ln n}{n-t}) \leq \exp(-2 \ln n) = \frac{1}{n^2}.$$

So with high probability the number of nodes in H is at most (we use the fact that $\sqrt[3]{n} \leq k \leq \sqrt{n}$ and that there exists a constant $0 < \beta < 1$ such that $t \leq \beta k$)

$$\begin{aligned} M &\leq \mathrm{E}[\Delta_t(w_t)] + s = (1-p)^{t-1}(k - t + p(n-k)) + \sqrt{n \ln n} \\ &\leq \sqrt{n} - t + \frac{\ln 2}{\sqrt[3]{n}}(n - \sqrt[3]{n}) + \sqrt{n \ln n} = O(n^{\frac{2}{3}}). \end{aligned}$$

Let $K^t = K \cap \Gamma_t(w_t)$, for $t+1 \leq i \leq k$ let X_i be the random variable indicating if $w_i \in \Gamma_t(w_t)$ (i.e. if $w_i \in K^t$) and $X = \sum_{i=t+1}^{k} X_i$ be the size of K^t. Then

$$\Pr[X_i = 1] = (1-p)^{t-1}$$

and

$$\mathrm{E}[X] = (1-p)^{t-1}(k - t - 1).$$

Using Chernoff bound

$$\Pr[X < \frac{1}{2}\mathrm{E}[X]] \leq \exp(-\frac{k-t-1}{8}) = \exp(-\Theta(k)).$$

So with high probability $G[\Gamma_t(w_t)]$ has a clique of size at least $\frac{1}{2}\mathrm{E}[X] = \frac{(1-p)^t}{2}(k-t-1) = \Theta(k) = \Omega(\sqrt{M})$. Since outside of K each edge appears with probability $\frac{1}{2}$ we get that also in $G[\Gamma_t(w_t)]$, each edge outside the clique K appears with probability $\frac{1}{2}$. And so with probability $\geq 1 - \frac{1}{n}$, $G[\Gamma_t(w_t)]$, is an instance of the hidden clique problem and has the required parameters.

Let $\widetilde{C}_t$ be the largest clique in $G[\Gamma_t(w_t)]$, we will show that $K^t = \widetilde{C}_t$. Since with high probability there is no clique of size $\Theta(k)$ in G outside K it must be that $K^t \subseteq \widetilde{C}_t$. We note that in the construction of $G[\Gamma_t(w_t)]$ we did not expose at its internal edges (we only used the fact that the nodes are connected to w_t with edge labeled t) and so as in G a node $u \notin K$ is connected to all of K^t with probability

$$(\frac{1}{2})^{|K^t|} = (\frac{1}{2})^{\Theta(k)} \leq \frac{1}{n^4}.$$

By a union bound, the probability that exists $u \notin K$ such that u is connected to all of K^t is at most $\frac{1}{n^3}$. For a node to be part of $\widetilde{C}_t$ it must connect to all of K^t and so with high probability there are no nodes from $V \setminus K$ in $\widetilde{C}_t$. So we conclude that with high probability the hidden clique of $G[\Gamma_t(w_t)]$ is $K^t = K \cap \Gamma_t(w_t)$ and we showed that $|K^t| = \Theta(k)$. □

Edges between nodes in H_{w_t} (as defined in the algorithm) and nodes from $V \setminus (K \cup \Gamma_t(w_t))$ were not exposed by the construction of H_{w_t} and of C_{w_t} (line 3) and so we can still think of these edges as random and that each appears with probability $\frac{1}{2}$. So for a fixed node $v \in V \setminus (K \cup \Gamma_t(w_t))$

$$\Pr[v \text{ is connected to all of } C_{w_t}] = (\frac{1}{2})^{|C(w_t)} = (\frac{1}{2})^{\Theta(k)} \leq \frac{1}{n^4}.$$

Thus

$$\Pr[\exists v \in V \setminus (K \cup \Gamma_t(w_t)) \text{ that is connected to all of } C_{w_t}] \leq \frac{1}{n^3}.$$

On the other hand, every $w \in K \setminus \Gamma_t(w_t)$ is connected to all of C_{w_t} since $C_{w_t} \subseteq K$. moreover by Claim 4.1 $K_{w_t} = (K \setminus \Gamma_t(w_t)) \cup C(w_t) = K$, and so if the algorithm gets to this iteration (i.e. when $u = w_t$) the algorithm will output K as needed. In any other iteration, if a clique of size k is found then with high probability it must be K since there are no other cliques of size k in G, and if no clique is found the algorithm simply moves on to the next iteration.

For algorithm OnePoint to fail at least one of the following must happen:

- The induced subgraph H_{w_t} does not have the parameters $(M, \frac{1}{2}, Q)$ with $Q = \Omega(\sqrt{M})$.
- There exists $v \notin K$ such that $v \in \widetilde{C}_t$ (where $\widetilde{C}_t$ is the hidden clique in H_{w_t}).

- The algorithm for finding the hidden clique in $H(w_t)$ in step 3 fails.
- For $u = w_t$, the completion in step 5 of H_u fails.

Each of these events happens with probability at most $\frac{1}{n}$, so overall the algorithm succeeds with probability at least $1 - \frac{4}{n}$. This conclude the proof of Theorem 1.4. □

4.2 Algorithm that uses many queries

We now show that given more temporal information, a logarithmic number of snapshots, the clique can be found if its size is polynomial (i.e. for every constant $\alpha > 0$).

Theorem 1.5. *Assuming $\alpha > 0$ is fixed, there exists an algorithm that given $\{E_t | t \in I\}$ and I for any set $I \subseteq [k]$ of size $\geq a \ln n$ for any constant $a \geq c \cdot 2^{\frac{10}{\alpha}+2}$ (for large enough constant c) finds K with probability at least $1 - \frac{5r}{n}$ ($r = r(\alpha, a)$ is a parameter to be determined later that does not depend on n), the algorithm runs in $n^{O(\frac{1}{\alpha})}$ time.*

Proof of Theorem 1.5. Let $b = \frac{15}{2\alpha}$ and c is a constant to be determined later. We present algorithm AnySteps that gets as input G, $\{E_t\}_{t \in I}$ and I and with high probability returns K. The algorithm partitions the graph to r (a constant to be chosen later) subgraphs $G_1, \ldots G_r$. For each $i \in [r]$ the algorithm finds M_i which is supposed to be a large fraction of $K \cap G_i$ by using Expand (line 4). These fractions of K are used to find all of K as follows. For every G_i look at a (non empty) M_j (for $j \neq i$), denote by Q_i all the nodes in G_i that are connected to all of M_j (line 6). The algorithm return the union of all sets Q_i. The intuition is that if M_i is a large enough fraction of K then with high probability only nodes from K will connect to it and so $Q_i = G_i \cap K$.

The algorithm uses procedure Expand that gets as input G_i a subgraph of G, $\{E_t\}_{t \in I}$, I and a threshold N. The procedure iterates over all sets of nodes S that form in G a clique of constant size b. The procedure tries to expand each such clique by looking for each $t \in I$ for a node u that connects to at least 0.4 fraction of S in E_t. If there are enough such nodes, i.e. at the end of the process, M the expanded set has at least N nodes, the result M is returned. The intuition is that if S is the subset of K of size b of nodes with the largest indexes in G_i, with high probability w_t will connect to a constant fraction of S in E_t and with high probability will be the only such node and so it will be added to M and if $|I| > N$ then $|M| > N$. So a large fraction of $K \cap G_i$ will be returned.

Algorithm AnySteps $(G, \{E_t\}_{t\in I}, I)$

for every node $v \in V$ **do**
    choose uniformly and independently at random $\sigma(v) \in \{1, \ldots, r\}$, denote by $V_i = \{v \in V | \sigma(v) = i\}$, $G_i = G[V_i]$ and $K_i = K \cap V_i$
for $i = 1, \ldots, r$ **do**
    $M_i \leftarrow$ Expand$(G_i, I, c \ln n)$
for $i = 1, \ldots, r$ **do**
    choose $j \neq i$ such that $M_j \neq \emptyset$ and set $Q_i = \{v \in V_i \mid v$ connects to all of M_j in $G\}$, if there is no such j set $Q_i = \emptyset$
$Q \leftarrow \bigcup_{i=1}^{r} Q_i$
return Q

Algorithm Expand $(G, \{E_t\}_{t\in I}, I, N)$

for every $S \subseteq V$, $|S| = b$ that is a clique in G **do**
    set $M \leftarrow S$
    for every $t \in I$ **do**
        if $\exists u \in V$ that is connected to at least 0.4 fraction of S in E_t **then**
            $M \leftarrow M \cup \{u\}$, if there is more than one, choose one arbitrarily
    if $|M| \geq N$ **then**
        return M
return $\emptyset$

It is easy to verify that the algorithm runs in $n^{b+O(1)} \leq n^{O(\frac{1}{\alpha})}$ time which is polynomial assuming $\alpha > 0$ is a constant.

Let $I_i = \{j \in I | w_j \in V_i\}$, $\mathrm{E}[|I_i|] = \frac{a \ln n}{r}$, taking $\rho = \sqrt{a} \ln n$, by Hoeffding's inequality

$$\Pr[||I_i| - \mathrm{E}[|I_i|]| \geq \rho] \leq \frac{1}{n^2}. \tag{1}$$

We denote by $\widehat{I_i} \subseteq I_i$ the set of b largest indices in I_i (note that b is a constant so by taking $r \leq \frac{1}{2}\sqrt{a}$ and by 1, with high probability $|I_i| \geq \sqrt{a} \ln n \gg \frac{15}{2\alpha} = b$). From now on we assume that for every $1 \leq i \leq r$ $||I_i| - \mathrm{E}[|I_i|]| \geq \rho$, note this indeed happens with probability $\geq 1 - \frac{r}{n^2}$

We denote by $L_i \subseteq K_i$ the b nodes of the clique that are in G_i with largest indexes, and by $\widehat{K}_i = \{w_j \in K | j \in I_i\}$ nodes of the clique that are in G_i that we have information on the step in which they act (i.e. connect to all other nodes in the clique).

The sets M_i

For each $1 \leq i \leq r$ we analyze the execution of Expand $(G_i, I, c \ln n)$, denoting by S_i the set S in this execution and analyzing the iteration of the outer loop of the procedure where $S_i = L_i$. For $v \in V_i \setminus K_i$ and $t \in I$

$$
\begin{aligned}
&\Pr[v \text{ is connected to at least } 0.4 \text{ fraction of } S \text{ in } E(G_i)_t] \leq \\
&\leq \sum_{l=0.4b}^{b} \sum_{R \subseteq S_i, |R|=l} \Pr[v \text{ is connected exactly to } R \text{ in } E(G_i)_t] \\
&\leq \sum_{l=0.4b}^{b} b^l ((1-p)^{t-1}p)^l (1-(1-p)^{t-1}p)^{b-l} \\
&\leq \sum_{l=0.4b}^{b} b^l (\frac{p}{2})^l \\
&\leq b^{b+1} (\frac{p}{2})^{0.4b} \\
&\leq b^{b+1} (\frac{\ln 2}{2n^\alpha})^{\frac{3}{\alpha}} \\
&\leq \frac{b^{b+1}}{n^3}.
\end{aligned}
$$

Using a union bound we get that with probability $\geq 1 - \frac{1}{n}$ no node $v \in V_i \setminus K_i$ will be added to M_i in step 4 in EXPAND $(G_i, I, c \ln n)$.

We call a time step t "good in part i" if $w_t \in \widehat{K}_i$ and w_t is connected to at least $0.4b$ nodes of S_i in $E(G_i)_t$. We let $Z_i(t)$ be the random variable indicating if step t is good in part i, and $Z_i = \sum_{t \in I} Z_i(t)$ the number of good steps in part i out of the

$$a \ln n - |I_i| \geq a \ln n - (\frac{a}{r} - \sqrt{a}) \ln n \geq (a - \sqrt{a}) \ln n$$

steps for which we have information, where the inequality happens due to our assumption that $||I_i| - \mathrm{E}[|I_i|]| \leq \rho$ and $r \leq \frac{1}{2}\sqrt{a}$. For every $t < l$ such that $w_t, w_l \in V_i$ we have $Pr((w_t, w_l) \in E(G_i)_t) = (1-p)^{t-1}$, we look at three cases:

- $\frac{1}{2} \leq (1-p)^{t-1} \leq \frac{3}{4}$:

$$
\begin{aligned}
\Pr[Z_i(t) = 1] &= \Pr[w_t \in \widehat{K}_i] \sum_{j=0.4b}^{b} \binom{b}{j} (1-p)^{(t-1)j} (1-(1-p)^{t-1})^{b-j} \\
&\geq \frac{1}{r} \sum_{j=0.4b}^{b} \binom{b}{j} (\frac{1}{2})^j (\frac{1}{4})^{b-j} \geq \frac{1}{r} (\frac{1}{2})^b.
\end{aligned}
$$

- $\frac{3}{4} < (1-p)^{t-1} \leq 1-p$:

$$
\begin{aligned}
\Pr[Z_i(t) = 1] &= \Pr[w_t \in \widehat{K}_i] \sum_{j=0.4b}^{b} \binom{b}{j} (1-p)^{(t-1)j} (1-(1-p)^{t-1})^{b-j} \\
&\geq \frac{1}{r} \sum_{j=0.4b}^{b} \binom{b}{j} (\frac{3}{4})^j p^{b-j} \geq \frac{1}{r} (\frac{3}{4})^b.
\end{aligned}
$$

- $(1-p)^{t-1} = 1$ i.e. $t = 1$:

$$\Pr[Z_i(1) = 1] = \Pr[w_1 \in \widehat{K}_i] = \frac{1}{r}.$$

So $\mathrm{E}[Z_i] \geq (\frac{1}{2})^b \frac{a-\sqrt{a}}{r} \ln n$. Using Chernoff bound, and taking $c = \frac{a-\sqrt{a}}{2^{b+1}r}$:

$$\Pr[Z_i \leq c \ln n] = \Pr[Z_i \leq \frac{1}{2}\mathrm{E}[Z_i]] \leq \exp(-\frac{\frac{1}{4}(\frac{1}{2})^b \frac{a-\sqrt{a}}{r} \ln n}{2})) = n^{-\frac{a-\sqrt{a}}{2^{b+3}r}}.$$

As long as $r \leq \frac{a-\sqrt{a}}{2^{b+3}}$ we get that the above probability is $\leq \frac{1}{n}$.

So with probability at least $1 - \frac{1}{n}$ at the beginning of step 6 of EXPAND, $|M_i| \geq c \ln n$ (for a fixed i). So, under all the above assumptions and using a union bound

$$\Pr[\exists 1 \leq i < j \leq r, |M_j|, |M_i| < c \ln n] = 1 - \frac{1}{n^r} - \frac{r}{n^{r-1}}(1 - \frac{1}{n}) \geq 1 - \frac{2}{n}.$$

Note that all the sets M_i are subsets of the clique K, since with high probability, no node $v \in V_i \setminus K_i$ is added to M_i.

The set Q

Assume that in order to build Q_i, the algorithm chose $j \neq i$ such that the first part of the algorithm succeeds in part j, i.e. $|M_j| \geq c \ln n$ (and so w.h.p M_j is part of the hidden clique K). We note that the cross-edges (the edges between two different parts) are independent of each other and of the choice of i, j and so, for $w_l \in V_i$, w_l is a neighbor of $c \ln n$ nodes of M_j in the graph G, and so $w_l \in Q_i$. On the other hand, for $v \in V_i \setminus K_i$, the expected degree of v towards any fixed set of size $c \ln n$ is $\frac{c}{2} \ln n$, using Hoeffding's inequality the probability that v have a degree of $\ln n$ relatively to M_j is $\leq \frac{1}{n^2}$. So with probability $\geq 1 - \frac{1}{n}$, no node that is not part of the clique will be added to Q_i and so we get that with this probability $Q = K$.

In total the probability of success is at least $1 - \frac{5r}{n}$, we can simply take $r = 2$. Besides requiring $r > 1$ in order to have crossover edges there is no lower bound on the value of r. This completes the proof of Theorem 1.5. □

References

[AKS98] N. Alon, M. Krivelevich, and B. Sudakov. Finding a large hidden clique in a random graph. *Random Structures and Algorithms*, 13(3-4):457–466, 1998. doi:10.1002/(SICI)1098-2418(199810/12)13:3/4<457::AID-RSA14>3.0.CO;2-W.

[AV11] B. P. Ames and S. A. Vavasis. Nuclear norm minimization for the planted clique and biclique problems. *Mathematical programming*, 129(1):69–89, 2011.

[BV09] S. C. Brubaker and S. S. Vempala. Random tensors and planted cliques. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 406–419. Springer, 2009.

[DGGP14] Y. Dekel, O. Gurel-Gurevich, and Y. Peres. Finding hidden cliques in linear time with high probability. *Combinatorics, Probability and Computing*, 23(01):29–49, 2014.

[DM13] Y. Deshpande and A. Montanari. Finding hidden cliques of size $\sqrt{\frac{n}{e}}$ in nearly linear time. *arXiv preprint arXiv:1304.7047*, 2013.

[Fei04] U. Feige. Approximating maximum clique by removing subgraphs. *SIAM Journal on Discrete Mathematics*, 18(2):219–225, 2004.

[FK00] U. Feige and R. Krauthgamer. Finding and certifying a large hidden clique in a semirandom graph. *Random Structures and Algorithms*, 16(2):195–208, 2000.

[FK03] U. Feige and R. Krauthgamer. The probable value of the Lovász–Schrijver relaxations for maximum independent set. *SIAM Journal on Computing*, 32(2):345–370, 2003.

[FK08] A. Frieze and R. Kannan. A new approach to the planted clique problem. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 2 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 187–198. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2008. doi:http://dx.doi.org/10.4230/LIPIcs.FSTTCS.2008.1752.

[FR10] U. Feige and D. Ron. Finding hidden cliques in linear time. 21st International Meeting on Probabilistic, Combinatorial, and Asymptotic Methods in the Analysis of Algorithms (AofA'10), 2010. Available from: http://www.dmtcs.org/dmtcs-ojs/index.php/proceedings/article/view/dmAM0114.

[GM75] G. R. Grimmett and C. J. McDiarmid. On colouring random graphs. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 77, pages 313–324. Cambridge Univ Press, 1975.

[Hs99] J. Hstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182(1):105–142, 1999. Available from: `http://dx.doi.org/10.1007/BF02392825`, `doi:10.1007/BF02392825`.

[Jer92] M. Jerrum. Large cliques elude the metropolis process. *Random Structures & Algorithms*, 3(4):347–359, 1992.

[Kar72] R. M. Karp. *Reducibility among combinatorial problems*. Springer, 1972.

[Kar76] R. M. Karp. The probabilistic analysis of some combinatorial search algorithms. In *Algorithms and complexity (Proc. Sympos.)*, pages 1–19. Academic Press, 1976.

[Kuč95] L. Kučera. Expected complexity of graph partitioning problems. *Discrete Applied Mathematics*, 57(2):193–212, 1995.

A Additional proofs for section 3

We begin by analyzing the random variables presented in section 3.

For the random variable Y_v^t we have:

- For $u, v \in V$ such that at least one of them is outside K:
$$\Pr[Y_v^t(u) = 1] = (1-p)^{t-1}p$$
- For $w_i, w_j \in K$ such that $i < t$:
$$\Pr[Y_{w_i}^t(w_j) = 1] = 0$$
- For $w_i, w_j \in K$ such that $i, j > t$:
$$\Pr[Y_{w_i}^t(w_j) = 1] = (1-p)^{t-1}p$$
- For $w_i \in K$ such that $i > t$:
$$\Pr[Y_{w_i}^t(w_t) = 1] = (1-p)^t$$
- For $w_i \in K$ such that $i < t$:
$$\Pr[Y_{w_i}^t(w_t) = 1] = 0$$
- For $u \notin K$:
$$\Pr[Y_u^t(w_t) = 1] = (1-p)^t p$$

For the random variable $Y_Q^S(u)$ we have:

- For $u \notin K$ $\Pr[Y_Q^S(u) = 1] = (1-p)^{\sum_{i=1}^m t_i - m} p^m$.
- For $u = w_j \in K$ and $Q = \{w_{i_1}, \ldots, w_{i_m}\}$, if exists $1 \leq l \leq m$ such that $t_l > i_l$ then $\Pr[Y_Q^S(u) = 1] = 0$.
- For $u = w_j \in K$, if $j < \max\{t_1, \ldots, t_m\}$ then $\Pr[Y_Q^S(u) = 1] = 0$.
- For $u = w_j \in K$, if $j > \max\{t_1, \ldots, t_m\}$ then
$$\Pr[Y_Q^S(u) = 1] = (1-p)^{\sum_{l=1}^m t_l - m} p^{m-m'}$$
Where $m' = |\{1 \leq l \leq m \mid v_l = w_{t_l}\}| \in \{0, \ldots, m\}$.
- For $u = w_j \in K$, if exists $1 \leq l \leq m$ such that $j = t_l$ then
$$\Pr[Y_Q^S(u) = 1] = (1-p)^{\sum_{l=1^m} t_l - m} p^{m-m'-1}$$
Where again $m' = |\{1 \leq l \leq m \mid v_l = w_{t_l}\}| \in \{0, \ldots, m-1\}$.

Proof of Lemma 3.5. We first calculate the different expectations:

$$\begin{aligned}
\mu_1 &= \mathrm{E}[Y^S_{Q_1}] = (n-k)(1-p)^{\sum_{l=1}^{M} t_i - m}p^M + (k-\rho)(1-p)^{\sum_{l=1}^{m} t_i - M}.\\
\mu_2 &= \mathrm{E}[Y^S_{Q_2}] = (n-m)(1-p)^{\sum_{l=1}^{M} t_i - m}p^M.\\
\mu_3 &= \mathrm{E}[Y^S_{Q_3}] = (n-k)(1-p)^{\sum_{l=1}^{M} t_i - m}p^M + (k-\rho-1)(1-p)^{\sum_{l=1}^{M} t_i - M}p^M + (1-p)^{\sum_{l=1}^{M} t_i - M}p^{M-1}\\
&= (n-\rho-1+\frac{1}{p})(1-p)^{\sum_{l=1}^{M} t_i - m}p^M.\\
\mu_4 &= \mathrm{E}[Y^S_{Q_4}] = (n-k)(1-p)^{\sum_{l=1}^{M} t_i - M}p^M.\\
\mathrm{E}[Y^S_{Q_5}] &= \sum_{u\in V}\mathrm{E}[\prod_{i=1}^{3} Y^{S_i}_{Q_i}(u)] = \sum_{u\in V}\prod_{i=1}^{3}\mathrm{E}[Y^{S_i}_{Q_i}(u)]\\
&= (n-k)(1-p)^{\sum_{l=1}^{M} t_l - M}p^M + \sum_{w_j\in K, j<\rho}(1-p)^{\sum_{l=m_1+1}^{m_2} t_l - m_2}p^{m_2}\cdot I_{\{m_2=M\}} +\\
&+ \sum_{w_j\in k, j>\rho+1}(1-p)^{\sum_{l=1}^{M} t_l - M}p^{m_2+m_3} + (1-p)^{\sum_{l=1}^{M} t_l - M}p^{m_2+m_3-I_{\{m_3>0\}}}\\
&= (n-k)(1-p)^{\sum_{l=1}^{M} t_l - M}p^M + (\rho-1)(1-p)^{\sum_{l=m_1+1}^{m_2} t_l - m_2}p^{m_2}\cdot I_{\{m_2=M\}} +\\
&+ (k-\rho-1+p^{-I_{\{m_3>0\}}})(1-p)^{\sum_{l=1}^{M} t_l - M}p^{m_1+m_3}.
\end{aligned}$$

Where:

- $I_{\{m_2=M\}} = 1$ if $m_2 = M$ and $I_{m_2=M} = 0$ otherwise.
- $I_{\{m_3>0\}} = 1$ if $m_3 > 0$ and $I_{m_3=0} = 0$ otherwise.

We consider a composite set, i.e. at least 2 of $m_1,\ m_2,\ m_3$ are larger than 0 we get:

$$\mu_5 = \mathrm{E}[Y^S_Q] = (n-k)(1-p)^{\sum_{l=1}^{m} t_l - m}p^m + (k-\rho-1+p^{-I_{m_3>0}})(1-p)^{\sum_{l=1}^{m} t_l - m}p^{m_2+m_3}$$

It's easy to see that since $m = \Theta(1)$ and $\rho = \Theta(k)$ $\mu_3, \mu_4 \leq \mu_2$ so we need to show $\mu_2 \leq \mu_5 \leq \mu_1$:

- To show that $\mu_5 \leq \mu_1$ we look at the difference and see if it's grater than 0:

$$\mu_1 - \mu_5 = (k-\rho)(1-p)^{\sum_{l=1}^{m} t_l - m} - (k-\rho-1+p^{-I_{m_3>0}})(1-p)^{\sum_{l=1}^{m} t_l - m}p^{m_2+m_3} \geq 0$$

$$k-\rho \geq (k-\rho-1+p^{-I_{m_3>0}})p^{m_2+m_3}$$

The left hand side is $\Theta(k)$ while the right hand side is $\Theta(k^{1-m_2-m_3})$, since we know that at least one of m_2, m_3 is grater than 1 we get that the right hand side is smaller, thus the inequality follows.

- To show that $\mu_2 \leq \mu_5$ we look at the difference and see if it's grater than 0:

  $$\mu_5 - \mu_2 = (k - \rho - 1 + p^{-I_{m_3>0}})(1-p)^{\sum_{l=1}^{m} t_l - m} p^{m_2+m_3} - (k-m)(1-p)^{\sum_{l=1}^{m} t_l - m} p^m \geq 0$$

  $$(k - \rho - 1 + p^{-I_{m_3>0}})p^{m_2+m_3} \geq (k-m)p^m$$

  The left hand side is $\Theta(k^{1-m_2-m_3})$ and the right hand side is $\Theta(k^{1-m})$. So if $m_2 + m_3 < m$ the inequality follows. If $m_2 + m_3 = m$, and since we are assuming that at least two of m_1, m_2, m_3 are grater than 0 we know that $m_3 > 0$, we can further simplify and get:

  $$m \geq \rho + 1 - \frac{1}{p}$$

  Since $\frac{\ln 2}{k+\ln 2} \leq p \leq \frac{\ln 2}{k}$ and $\frac{k}{3} + 1 \leq \rho \leq \frac{2k}{3}$, the inequality follows also in this case.

□