On Finding Dense Subgraphs *

Samir Khuller and Barna Saha

University of Maryland College Park
{samir,barna}@cs.umd.edu

Abstract. Given an undirected graph $G = (V, E)$, the density of a subgraph on vertex set S is defined as $d(S) = \frac{|E(S)|}{|S|}$, where $E(S)$ is the set of edges in the subgraph induced by nodes in S. Finding subgraphs of maximum density is a very well studied problem. One can also generalize this notion to directed graphs. For a directed graph one notion of density given by Kannan and Vinay [12] is as follows: given subsets S and T of vertices, the density of the subgraph is $d(S,T) = \frac{|E(S,T)|}{\sqrt{|S||T|}}$, where $E(S,T)$ is the set of edges going from S to T. Without any size constraints, a subgraph of maximum density can be found in polynomial time. When we require the subgraph to have a specified size, the problem of finding a maximum density subgraph becomes NP-hard. In this paper we focus on developing fast polynomial time algorithms for several variations of dense subgraph problems for both directed and undirected graphs. When there is no size bound, we extend the flow based technique for obtaining a densest subgraph in directed graphs and also give a linear time 2-approximation algorithm for it. When a size lower bound is specified for both directed and undirected cases, we show that the problem is NP-complete and give fast algorithms to find subgraphs within a factor 2 of the optimum density. We also show that solving the densest subgraph problem with an upper bound on size is as hard as solving the problem with an exact size constraint, within a constant factor.

1 Introduction

Given an undirected graph $G = (V, E)$, the density of a subgraph on vertex set S is defined as $d(S) = \frac{|E(S)|}{|S|}$, where $E(S)$ is the set of edges in the subgraph induced by S. The problem of finding a densest subgraph of a given graph G can be solved optimally in polynomial time, despite the fact that there are exponentially many subgraphs to consider [16, 11]. In addition, Charikar [6] showed that we can find a 2 approximation to the densest subgraph problem in linear time using a very simple greedy algorithm (the greedy algorithm was previously studied by Asahiro et. al. [4]). This result is interesting because in many applications of analyzing social networks, web graphs etc., the size of the graph involved could be very large and so having a fast algorithm for finding an approximately dense subgraph is extremely useful. However when there is a size constraint specified - namely find a densest subgraph of exactly k vertices (*DkS*), the densest k subgraph problem becomes NP-hard [8, 3]. When $k = \Theta(|V|)$, Asahiro et. al. [4] gave a constant factor approximation algorithm for the *DkS* problem. However for

* Research supported by NSF CCF 0728839 and a Google Research Award.

general k, the algorithm developed by Feige, Kortsarz and Peleg [8] achieves the best approximation guarantee of $O(n^a)$, where $a < \frac{1}{3}$. Khot [13] showed that there does not exist any PTAS for the *DkS* problem under a reasonable complexity assumption. Closing the gap between the approximation factor and the hardness guarantee is an important open problem.

Recently, Andersen and Chellapilla [2] considered two variations of the problem of finding a densest k subgraph. The first problem, the densest at-least-k-subgraph problem (*DalkS*) asks for an induced subgraph of highest density among all subgraphs with at least k nodes. This relaxation makes *DalkS* significantly easier to approximate and Andersen et.al. gave a fast algorithm based on Charikar's greedy algorithm that guarantees a 3 approximation for the *DalkS* problem. In addition, Andersen [1] showed that this problem has a polynomial time 2 approximation, albeit with significantly higher running time. However it was left open as to whether or not this problem is NP-complete. The second problem studied was the densest at-most-k-subgraph problem (*DamkS*), which asks for an induced subgraph of highest density among all subgraphs with at most k nodes. For the *DamkS* problem, Andersen et.al. showed that if there exists an α approximation for *DamkS*, then there is a $\Theta(\alpha^2)$ approximation for the *DkS* problem, indicating that this problem is likely to be quite difficult as well.

For directed graphs, Kannan and Vinay [12] defined a suitable notion of density to detect highly connected subgraphs and provided a $\Theta(\log n)$ approximation algorithm for finding such dense components. Let $G = (V, E)$ be a directed graph and S and T be two subsets of nodes of V. Density corresponding to S and T is defined as $d(S,T) = \frac{|E(S,T)|}{\sqrt{|S||T|}}$, where $E(S,T)$ consists of the edges going from S to T. Charikar [6] showed that the problem can be solved in polynomial time by solving an LP using n^2 different values of a parameter. However a max-flow based technique similar to the one developed by Goldberg [11] for the densest subgraph problem in undirected graphs was not known for directed graphs. It was mentioned as one of the open problems in [6]. In addition to providing a polynomial time solution for the densest subgraph problem in directed graphs, Charikar also gave a 2 approximation algorithm which runs in $O(|V|^3 + |V|^2|E|)$ time.

Densest subgraph problems have received significant attention for detecting important substructures in massive graphs like web and different social networks. In a web graph, hubs (resource lists) and authorities (authoritative pages) on a topic are characterized by large number of links between them [15]. Finding dense subgraphs also acts as a useful primitive for discovering communities in web and social networks, for compressed representation of a graph and for spam detection [7, 5, 10]. Gibson et. al. [10] provided effective heuristics based on two-level fingerprints for finding large dense subgraphs in massive graphs. Their aim was to incorporate this step into web search engine for link spam control. Dourisboure gave a scalable method for identifying small dense communities in web graph [7]. Buehrer showed how large dense subgraphs can be useful in web graph compression and sub-sampling a graph [5]. In all these applications the underlying graph is massive and thus fast scalable algorithms for detecting dense subgraphs are required to be effective.

One of the main new insights in this paper is to illustrate the power of the flow based methods [11, 16] to find dense subgraphs not only when there is no requirement on the

size of the obtained subgraph, but also for cases when there is a constraint on the size of the obtained subgraph. Precisely our contributions are as follows:

1.1 Contributions

- For the densest subgraph problem without any size restrictions (Section 2):
  - We give a max-flow based polynomial time algorithm for solving the densest subgraph problem in directed graphs.
  - We give a linear time 2-approximation algorithm for the densest subgraph problem in directed graphs.
- For the densest at least k subgraph problem (Section 3):
  - We show that the densest at least k subgraph problem is NP-Hard.
  - For undirected graphs, we give a flow-based and LP based approximation algorithms, for the densest at least k subgraph problem. These run much faster than the polynomial time approximation algorithm of Andersen and deliver the same worst case approximation factor of 2.
  - We define the notion of densest at least k_1, k_2 subgraph problem for directed graphs and give a 2-approximation algorithm for it.
- Densest at most k subgraph problem (Section 4):
  - We show that approximating the densest at most k subgraph problem is as hard as the densest k subgraph problem within a constant factor, specifically an α approximation for *DamkS*, implies a 4α approximation for *DkS*.

2 Densest subgraph without any size restriction

In this section, first we give a max-flow based algorithm for the densest subgraph problem in directed graphs. For undirected graphs, Goldberg [11] developed a flow based algorithm, that finds a densest subgraph in polynomial time. However, for directed graphs, no flow based algorithm was known. Next we consider the greedy algorithm for densest subgraph in undirected graphs proposed by Charikar [6] and develop an extension of this algorithm to give a 2 approximation algorithm for finding a densest subgraph in directed graphs. This improves the running time from $O(|V|^3 + |V|^2|E|)$ to $O(|V| + |E|)$. We also give a very simple proof of 2-approximation for the greedy algorithm developed by Charikar [6] to obtain a densest subgraph in undirected graphs.

2.1 Max-flow based algorithm for finding densest subgraphs in directed graphs

For a directed graph $G = (V, E)$, we wish to find two subsets of nodes S and T, such that $d(S,T) = \frac{|E(S,T)|}{\sqrt{|S||T|}}$ is maximized. Let us denote the optimum subsets of nodes by S^* and T^* respectively. To detect such subsets of nodes, we first guess the value of $\frac{|S^*|}{|T^*|}$ in the optimum solution. Since there are $|V|^2$ possible values, in $\Theta(|V|^2)$ time, it is possible to guess this ratio exactly[1]. Let this ratio be a. We create a bipartite graph

[1] If we want $(1+\epsilon)$ approximation, only $O(\frac{\log |V|}{\epsilon})$ guessed values suffice.

$G' = (V_1, V_2, E)$, where $V_1 = V_2 = V$ and for every directed edge (i, j) in the original graph, we add an edge from vertex $i \in V_1$ to $j \in V_2$. We now wish to find $S \subseteq V_1$ and $T \subseteq V_2$, such that $\frac{|E(S,T)|}{\sqrt{|S||T|}}$ is maximized. We also know, $\frac{|S^*|}{|T^*|} = a$.

We add a source s and a sink t to $G' = (V_1, V_2, E)$. We guess the value of the optimum (maximum) density. Let our guessed value be g. The following edges with weights are then inserted into $G' = (V_1, V_2, E)$:

- We add an edge of weight m from source s to each vertex of V_1 and V_2, where $m = |E|$.
- We add an edge of weight $(m + \frac{g}{\sqrt{a}})$ from each vertex of V_1 to the sink t.
- We add an edge from each vertex j of V_2 to sink t of weight $m + \sqrt{a}g - 2d_j$, where d_j is the in-degree of j.
- All the edges going from V_1 to V_2 are given weight 0. For each edge going from V_1 to V_2, a reverse edge of weight 2 is added.

Now consider a s-t min-cut in this weighted graph. Since the cut $\{s\}, \{t, V_1, V_2\}$ has weight $m(|V_1| + |V_2|)$, the min-cut value is $\leq m(|V_1| + |V_2|)$. Now consider the cut $\{s, S \subseteq V_1, T \subseteq V_2\}$,$\{t, (V_1 \setminus S) \subseteq V_1, (V_2 \setminus T) \subseteq V_2\}$. The number of edges crossing the cut is,

$$m(|V_1| - |S| + |V_2| - |T|) + (m + \frac{g}{\sqrt{a}})|S| + \sum_{i \in T}(m + \sqrt{a}g - 2d_i) + \sum_{\substack{i \in T, j \in V_1 - S, \\ (j,i) \in E(G)}} 2$$

$$= m(|V_1| + |V_2|) + |S|\frac{g}{\sqrt{a}} + |T|\sqrt{a}g - 2|E(S,T)|$$

$$= m(|V_1| + |V_2|) + \frac{|S|}{\sqrt{a}}\left(g - \frac{|E(S,T)|}{|S|/\sqrt{a}}\right) + |T|\sqrt{a}\left(g - \frac{|E(S,T)|}{|T|\sqrt{a}}\right)$$

Let us denote the optimum density value by d_{OPT}. If $g < d_{OPT}$, then there exists S and T (corresponding to the optimum solution), such that both $\left(g - \frac{|E(S,T)|}{|S|/\sqrt{a}}\right)$ and $\left(g - \frac{E(S,T)}{|T|\sqrt{a}}\right)$ are negative. Therefore S and T are nonempty. If the guessed value $g > d_{OPT}$, let if possible S and T be non-empty. Let in this returned solution, $\frac{|S|}{|T|} = b$. We have,

$$\begin{aligned}
&\frac{|S|}{\sqrt{a}}\left(g - \frac{|E(S,T)|}{|S|/\sqrt{a}}\right) + |T|\sqrt{a}\left(g - \frac{E(S,T)}{|T|\sqrt{a}}\right) \\
&= \sqrt{|S||T|}\frac{\sqrt{b}}{\sqrt{a}}\left(g - \frac{d(S,T)}{\sqrt{b}/\sqrt{a}}\right) + \sqrt{|S||T|}\frac{\sqrt{a}}{\sqrt{b}}\left(g - \frac{d(S,T)}{\sqrt{a}/\sqrt{b}}\right) \\
&= \sqrt{|S||T|}\left(\left(\frac{\sqrt{b}}{\sqrt{a}} + \frac{\sqrt{a}}{\sqrt{b}}\right)g - 2d(S,T)\right) \qquad (1)
\end{aligned}$$

Now, $\left(\frac{\sqrt{b}}{\sqrt{a}} + \frac{\sqrt{a}}{\sqrt{b}}\right) \geq 2$ $\forall$ reals a, b and we have, $g > d_{OPT} \geq d(S,T)$. Hence the value of (1) is > 0. Thus if S and T are non-empty, then this cut has value $>$

$m(|V_1| + |V_2|)$. Hence if $g > d_{OPT}$, min-cut is $(\{s\}, \{t, V_1, V_2\})$. If the guessed value $g = d_{OPT}$, then we get a cut of the same cost as the trivial min-cut, even by having S and T corresponding to S^* and T^* respectively. We can always ensure that we obtain a min-cut, which has the biggest size on the source side. Thus when the guessed value is correct, the optimum subsets S and T are obtained from the subsets of vertices of V_1 and V_2 that belong to the side of the cut that contains s. The algorithm detects the correct value of g using a binary search, similar to Goldberg's algorithm for finding a densest subgraph in undirected graphs [11]. Also it is easy to verify that, when the correct value of g is guessed, we have $b = a$. Using a parametric max-flow algorithm [9], the total time required is same as one flow computation within a constant factor.

2.2 2 approximation algorithm for the densest subgraph problem in undirected and directed graphs

We first consider Charikar's greedy algorithm [6] for densest subgraphs in undirected graphs. The greedy algorithm at each step chooses a vertex of minimum degree, deletes it and proceeds for $(n-1)$ steps, where $|V| = n$. At every step the density of the remaining subgraph is calculated and finally the one with maximum density is returned.

Algorithm 2.1: DENSEST-SUBGRAPH($G = (V, E)$)

$n \leftarrow |V|, H_n \leftarrow G$
for $i = n$ *to* 2
do $\begin{cases} \text{Let } v \text{ be a vertex in } H_i \text{ of minimum degree} \\ H_{i-1} \leftarrow H_i - \{v\} \end{cases}$
return $(H_j,$ which has the maximum density among $H_i's, i = 1, 2, .., n)$

We show that the above greedy algorithm *Densest-Subgraph* achieves an approximation factor of 2 for undirected networks. This is not a new result. However our proof is simpler than the one given by Charikar. For directed graphs, Charikar developed a different greedy algorithm, that has a significantly higher time-complexity of $O(|V|^3 + |V|^2|E|)$. We show that the algorithm *Densest-Subgraph-Directed*, which is a generalization of the algorithm *Densest-Subgraph* detects a subgraph, with density within a factor of 2 of the optimum for directed graphs. This reduces the time complexity from $O(|V|^3 + |V|^2|E|)$ to $O(|V| + |E|)$.

Theorem 1. *The greedy algorithm* **Densest-Subgraph** *achieves a* 2*-approximation for the densest subgraph problem in undirected networks.*

Proof. Let $d_{OPT} = \lambda$. Observe that in an optimum solution, every vertex has degree $\geq \lambda$. Otherwise removing a vertex of degree $< \lambda$, will give a subgraph with higher density. Consider the iteration of the greedy algorithm when the first vertex of the optimum solution is removed. At this stage all the vertices in the remaining subgraph have degree $\geq \lambda$. If the number of vertices in the subgraph is s, then the total number of edges is $\geq \lambda s/2$, and the density is $\geq \lambda/2$. Since the greedy algorithm returns the subgraph with the highest density over all the iterations, it always returns a subgraph with density at least $\frac{1}{2}$ of the optimum. □

With a little work, one can make examples showing that the bound of 2 is tight for Charikar's algorithm (details omitted).

We now consider the case of directed graphs. In a directed graph, for each vertex we count its in-degree and out-degree separately. Let v_i be a vertex with minimum in-degree and v_o be a vertex with minimum out-degree. Then we say v_i has minimum degree, if the in-degree of v_i is at most the out-degree of v_o, else v_o is said to have the minimum degree. In the first case, the vertex with minimum degree belongs to the category IN. In the second case, it belongs to the category OUT. The greedy algorithm for directed graphs deletes the vertex with minimum degree and then depending on whether it is of category IN or OUT, either deletes all the incoming edges or all the outgoing edges incident on that vertex, respectively. If the vertex becomes a singleton, the vertex is deleted. To compute the density of the remaining graph after an iteration of *Densest-Subgraph-Directed*, any vertex that has nonzero out-degree is counted in the S side and all the vertices with non-zero in-degree are counted in the T side. Therefore the same vertex might appear both in S and T and will be counted once in S and once in T. We denote the optimum solution by (S^*, T^*).

Algorithm 2.2: DENSEST-SUBGRAPH-DIRECTED($G = (V, E)$)

$n \leftarrow |V|, H_{2n} \leftarrow G, i \leftarrow 2n$
while $H_i \neq \emptyset$
do
- Let v be a vertex in H_i of minimum degree
- **if** category(v) $= IN$
  - **then** Delete all the incoming edges incident on v
  - **else** Delete all the outgoing edges incident on v
- **if** v has no edges incident on it **then** Delete v
- Call the new graph $H_{i-1}, i \leftarrow i - 1$

return (H_j which has the maximum density among $H_i's$)

Define $\lambda_i = |E(S^*, T^*)| \left(1 - \sqrt{1 - \frac{1}{|T^*|}}\right)$ and $\lambda_o = |E(S^*, T^*)| \left(1 - \sqrt{1 - \frac{1}{|S^*|}}\right)$.

Lemma 1. *In an optimal solution, each vertex in S^*, has out-degree $\geq \lambda_o$ and each vertex in T^* has in-degree $\geq \lambda_i$.*

Proof. Suppose if possible, $\exists v \in S^*$ with out-degree $< \lambda_o$. Remove v from S^*. The density of the remaining subgraph is $> \frac{E(S^*,T^*)-\lambda_o}{\sqrt{(|S^*|-1)|T^*|}} = d_{OPT}$, which is not possible. Similarly, every vertex $v \in T^*$ has degree $\geq \lambda_i$. □

Theorem 2. *The greedy algorithm* **Densest-Subgraph-Directed** *achieves a 2 approximation for the densest subgraph problem in directed networks.*

Proof. Consider the iteration of the greedy algorithm, when the vertices in S have out-degree $\geq \lambda_o$ and the vertices in T have in-degree $\geq \lambda_i$. Let us call the set of vertices on the side of S and T by S' and T' respectively. Then the number of edges, $E' \geq |S'|\lambda_o$, and also $E' \geq |T'|\lambda_i$. Hence, the density $d(S', T') \geq \sqrt{\frac{|S'|\lambda_o|T'|\lambda_i}{|S'||T'|}} = \sqrt{\lambda_o\lambda_i}$. Substituting the values of λ_o and λ_i from Lemma 1, we get $d(S', T')^2 \geq$

$|E(S^*,T^*)|^2\left(1-\sqrt{1-\frac{1}{|S^*|}}\right)\left(1-\sqrt{1-\frac{1}{|T^*|}}\right)$. Now putting $|S^*|=\frac{1}{\sin^2\theta}$ and $|T^*|=\frac{1}{\sin^2\alpha}$, we get $d(S',T')\geq\frac{|E(S^*,T^*)|}{\sqrt{|S^*||T^*|}}\frac{\sqrt{(1-cos\theta)(1-cos\alpha)}}{sin\theta sin\alpha}=\frac{d_{OPT}}{2cos\frac{\theta}{2}cos\frac{\alpha}{2}}\geq\frac{d_{OPT}}{2}$. □

3 Densest at least k subgraph problem

For undirected graphs, the *DalkS* algorithm tries to find a subgraph of highest density among all subgraphs, that have size $\geq k$. We prove that the *DalkS* problem is NP-complete. and develop two algorithms; a combinatorial algorithm and one based on solving a linear programming formulation of the *DalkS* problem. Each algorithm achieves an approximation factor of 2. Finally we consider the *DalkS* problem in directed graphs, and give a 2-approximation algorithm for the problem.

Theorem 3. DalkS *is NP-Hard.*

Proof. We reduce the densest k subgraph problem (this problem is NP-hard [8, 3]) to densest at least k subgraph problem. The entire proof can be found in [14]. □

We develop two algorithms for *DalkS* that both achieve an approximation factor of 2. We note that Andersen [1] proposed a 2 approximation algorithm, that requires n^3 max-flow computations. Even using the parametric flow computation [9] the running time is within a constant factor of n^2 flow computations. Whereas our first algorithm uses at most $max(1,(k-\gamma))$ flow computations using parametric flow algorithm and in general much less than that. Here γ is the size of the densest subgraph without any size constraint. The second algorithm is based on a linear programming formulation for *DalkS* and requires only a single solution of a LP.

3.1 Algorithm 1: Densest at least k subgraph

Let H^* denote the optimum subgraph and let d^* be the optimum density. The algorithm starts with the original graph G as G_0, and D_0 as $\emptyset$. In the ith iteration, the algorithm finds the densest subgraph H_i from G_{i-1} without any size constraint. If $|V(D_{i-1})|+|V(H_i)|\geq k$, the algorithm stops. Otherwise the algorithm adds H_i to D_{i-1} to obtain D_i. All the edges and the vertices of H_i are removed from G_{i-1}. For every vertex $v\in G_{i-1}\setminus H_i$, if v has l edges to the vertices in H_i, then in G_i a self loop of weight l is added to v. The algorithm then continues with G_i. When the algorithm stops, each subgraph D_i is padded with arbitrary vertices to make their size k. The algorithm then returns the D_j with maximum density.

Algorithm 3.1: DENSEST AT LEAST-K(G,k)

$D_0\leftarrow\emptyset, G_0\leftarrow G, i\leftarrow 1$
while $|V(D_i)|<k$
do $\begin{cases} H_i\leftarrow \text{ maximum-density-subgraph}(G_{i-1}) \\ D_i\leftarrow D_{i-1}\cup H_i \\ G_i=\text{ shrink}(G_{i-1},H_i), i\leftarrow i+1 \end{cases}$
for each D_i
do Add an arbitrary set of $max(k-|V(D_i)|,0)$ vertices to it to form D_i'
return $(D_j',$ which has the maximum density among the $D_i's)$

We prove that algorithm *Densest At least-k* achieves an approximation factor of 2.

Theorem 4. *The algorithm* **Densest At least-k** *achieves an approximation factor of* 2 *for the* DalkS *problem.*

Proof. If the number of iterations is 1, then H_1 is the maximum density subgraph of the original graph whose size is $\geq k$. Therefore $H^* = H_1$ and the algorithm returns it. Otherwise, say the algorithm iterates for $l \geq 2$ rounds. There can be two cases:

Case 1: *There exists a* $l' < l$ *such that* $E(D_{l'-1}) \cap E(H^*) \leq \frac{E(H^*)}{2}$ *and* $E(D_{l'}) \cap E(H^*) \geq \frac{E(H^*)}{2}$.

Case 2: *There exists no such* $l' \leq l$.

For case 2, we have for any $j \leq l-1$, $E(D_j) \cap E(H^*) \leq \frac{E(H^*)}{2}$. Therefore, $E(G_j) \cap E(H^*) \geq \frac{E(H^*)}{2}$. Consider $V' = V(G_j) \cap V(H^*)$. The density of the subgraph induced by V' in G_j is $\geq \frac{E(G_j) \cap E(H^*)}{|V'|} \geq \frac{E(H^*)}{2V(H^*)} = d^*/2$. Hence the density of H_l must be $\geq d^*/2$. So in case 1, for each $j \leq l$, the density of H_j is $\geq d^*/2$. Therefore the total number of edges in the subgraph D_l is $\geq \frac{d^* \sum_{j=1}^{l'} |V(H_j)|}{2}$, or the density of $D_{l'}$ is $\geq d^*/2$ and it has $\geq k$ vertices.

For case 1, the subgraph $D_{l'}$ has at least $E(H^*)/2$ edges and since $V(D_{l'}) \leq k$, the density of $D'_{l'}$ is $\geq \frac{d^*}{2}$.

Since the algorithm returns the subgraph D'_j with maximum density among all the D'_is, the returned subgraph has density at least $d^*/2$. □

There are example of graphs (see the extended version [14]) over which the approximation factor of $\frac{1}{2}$ is tight for algorithm *Densest At least-k Subgraph*.

3.2 Algorithm 2: Densest at least *k* subgraph

Next we give a LP based solution for the *DalkS* problem. Define a variable $x_{i,j}$ for every edge $(i,j) \in E(G)$ and a variable y_i for every vertex $i \in V(G)$. Consider now the following LP:

$$\text{maximize} \quad \sum_{i,j} x_{i,j} \tag{2}$$

$$x_{i,j} \leq y_i \ , \forall (i,j) \in E(G); \ x_{i,j} \leq y_j \ , \forall (i,j) \in E(G)$$

$$\sum_i y_i = 1; \ \ y_i \leq \frac{1}{l} \ , \forall i \in V(G); \ x_{i,j}, y_i \geq 0 \ , \forall (i,j) \in E(G), \forall i \in V(G)$$

Here $l \geq k$ is the size of the optimum solution of the *DalkS* problem. Since there can be $n-k+1$ possible sizes of the optimum solution, we can guess this value, putting different values for l. In Section 3.3 we show that by first running the algorithm *Densest-Subgraph* and then solving one single LP, we can guarantee a 2-approximation.

Lemma 2. *The optimum solution of LP (2) is greater than or equal to the optimum value of* DalkS.

Proof. Let the optimum solution for *DalkS* be obtained for a subgraph H having $l \geq k$ vertices and density λ. Consider a solution for the above LP, where each of the variables y_i corresponding to the vertices of H have value $\frac{1}{l}$. All the variable $x_{i,j}$ corresponding to the induced edges of H have value $\frac{1}{l}$. The solution is feasible, since it satisfies all the constraints of LP (2). The value of the objective function of the LP is $\sum_{(i,j)\in H} x_{i,j} = \frac{E(H)}{l} = \frac{E(H)}{V(H)} = \lambda$. Therefore the optimum value of the LP is $\geq \lambda$ □

Lemma 3. *If the value of the optimum solution of LP (2) is λ, a subgraph of size $\geq k$ with density $\geq \lambda/2$ can be constructed from that solution of LP (2).*

The proof can be found in the extended version [14]. The key idea is to show that there exists a value of $r \in [0, 1]$, such that if we consider the subgraph induced by the vertices with y value $\geq r$, then either it has density $\geq \lambda/2$ and size $\geq k$, or its size is $< k$, but it has more than half the number of edges the optimum solution has. In the later case, we can add arbitrary vertices to increase the size of the subgraph to k.

Theorem 5. *If the value of the optimum solution of LP (2) is λ, a subset S of vertices can be computed from the optimum solution of LP (2), such that*

$$d(G(S)) \geq \lambda \text{ and } |S| \geq k$$

Proof. Consider every possible subgraph by setting $r = y_i$ for all distinct values of y_i. By Lemma 3, there exists a value of r such that $|S(r)| \geq k$ and $\frac{|E(r)|}{|S(r)|} \geq v/2$, where v is an optimum solution of the LP. By Lemma 2 , $v \geq \lambda$ and hence the proof. □

The integrality gap of LP 2 is at least $\frac{5}{4}$. Also the approximation factor of $\frac{1}{2}$ is tight (see the extended version [14]).

3.3 Reducing the number of LP solutions

To reduce the number of LP solutions, we first run the algorithm *Densest-Subgraph*, consider the solutions over all the iterations that have $> k$ vertices and obtain the one with maximum density. We call this modified algorithm *Densest-Subgraph*$_{>k}$. We compare the obtained subgraph from *Densest-Subgraph*$_{>k}$ with the solution returned by the LP based algorithm with $l = k$. The final solution is the one which has higher density.

When the optimum solution for *DalkS* has exactly k vertices, Theorem 5 guarantees that we obtain a 2 approximation. Otherwise, the optimum subgraph has size $> k$. In this situation, the following lemma shows that the solution returned by *Densest-Subgraph*$_{>k}$ has density at least $\frac{1}{2}$ of the optimum solution of *DalkS*. Therefore using only a single solution of LP 2 along with the linear time algorithm *Densest-Subgraph*$_{>k}$, we can guarantee a 2 approximate solution for *DalkS*.

Lemma 4. *If the optimum subgraph of* DalkS *problem has size $> k$, then* **Densest-Subgraph**$_{>k}$ *returns a 2 approximate solution.*

Proof. Let the optimum density be λ. Since the size of the optimum solution is $> k$, if there exists any vertex in the optimum solution with degree $< \lambda$, then removing that we would get higher density and the size of the subgraph still remains $\geq k$. Hence all the vertices in the optimum solution has degree $\geq \lambda$. Now from Theorem 1, we get the required claim. □

3.4 Densest at least k subgraph problem for directed graphs

Given a directed graph $G = (V, E)$ and integers k_1, k_2, the densest at least k directed subgraph (*DaLkDS*) *problem finds two subsets of nodes S and T containing at least k_1 and k_2 vertices respectively for which $\frac{E(S,T)}{\sqrt{|S||T|}}$ is maximized.*

In this section we give a 2 approximation algorithm for the *DaLkDS* problem. Since there are two parameters, k_1, k_2; we refer to this problem by *densest at least-k_1, k_2 problem* from now on.

3.5 Densest at least k_1, k_2 directed subgraph problem

Let S^*, T^* represent the optimum solution of *DaLkDS* and d^* represent the value of the density corresponding to S^*, T^*. Let the ratio $\frac{|S^*|}{|T^*|}$ be a. Since the possible values of a can be $\frac{i}{j}$, where $i \geq k_1, j \geq k_2$ and $i, j \leq |V|$, we can guess the value of a. We run the max-flow based algorithm of Section 2.1 (maximum-directed-density-subgraph) with the chosen a to obtain the densest directed subgraph without any size constraints. Instead of shrinking and removing the vertices and the edges in the densest directed subgraph, as in algorithm *Densest At least-k* for *DalkS*, we only remove the edges and maintain the vertices. We continue this procedure for the same choice of a, until at some round both the sizes of S and T thus obtained exceed k_1 and k_2 respectively. Let S_i and T_i be the partial subsets of vertices obtained up to the ith round. We append arbitrary vertices A and B to S_i and T_i to form S'_i and T'_i respectively, such that $|S'_i| \geq k_1$ and $|T'_i| \geq k_2$. The algorithm returns S'_j, T'_j, such that $d(S'_j, T'_j)$ is maximum over all the iterations.

Algorithm 3.2: DENSEST AT LEAST-k_1, k_2(G, k_1, k_2, a)

$S_0 \leftarrow \emptyset, T_0 \leftarrow \emptyset, G_0 \leftarrow G, i \leftarrow 1$
while $|S_{i-1}| < k_1$ or $|T_{i-1}| < k_2$
do $\begin{cases} H_i(S,T) \leftarrow \text{ maximum-directed-density-subgraph}(G_{i-1}, a) \\ S_i \leftarrow S_{i-1} \cup H_i(S) \\ T_i \leftarrow T_{i-1} \cup H_i(T) \\ G_i = \text{ shrink}(G_{i-1}, H_i), i \leftarrow i+1 \end{cases}$
for each S_i, T_i
do $\begin{cases} \text{Add arbitrary } max(k_1 - |S_i|, 0) \text{ vertices to } S_i \text{ to form } S'_i \\ \text{Add arbitrary } max(k_2 - |T_i|, 0) \text{ vertices to } T_i \text{ to form } T'_i \end{cases}$
return (S'_j, T'_j) which has maximum density among the $(S'_i, T'_i)s$

Theorem 6. *Algorithm* **Densest At least-k_1, k_2** *achieves an approximation factor of* 2 *for the* DaLkDS *problem.*

Proof. For a chosen a, algorithm *Densest At least-k_1, k_2* returns subsets $H_i(S)$ and $H_i(T)$ at iteration i, such that $\frac{|H_i(S)|}{|H_i(T)|} = a$. Suppose up to l_1th iteration, $|S_{l_1}| < k_1$ and $|T_{l_1}| < k_2$. Let $|S_{l_1+1}| \geq k_1$, but up to l_2th iteration, $|T_{l_2}| < k_2$. At iteration $l_2 + 1$, $|T_{l_2+1}| \geq k_2$. Now we consider the following cases,

Case 1: $|E(S_{l_1}, T_{l_1})| \geq |E(S^*, T^*)|/2$.

Case 2: $|E(S_{l_2}, T_{l_2}) \bigcap E(S^*, T^*)| \leq |E(S^*, T^*)|/2$.

Case 3: $\exists l', l_1 < l' \leq l_2$, *such that* $|E(S_{l'}, T_{l'})| > |E(S^*, T^*)|/2$ *and* $|E(S_{l'-1}, T_{l'-1})| \leq |E(S^*, T^*)|/2$.

These three cases are mutually exclusive and exhaustive. When case 1 occurs, we can append arbitrary vertices to S_{l_1} and T_{l_1} to make their sizes respectively k_1 and k_2. In that case $\frac{E(S'_{l_1}, T'_{l_1})}{\sqrt{|S'_{l_1}||T'_{l_1}|}} \geq \frac{E(S^*, T^*)}{2\sqrt{|S^*||T^*|}} = d^*/2$. When case 2 occurs, at iteration l_2 at least half of the edges of the optimum are still not covered. Since no vertices are ever deleted, choice of S^* and T^* maintains the ratio a and returns a density which is at least $\frac{d^*}{2}$. Then we have $\forall i = 1, 2, .., l_2$, $\frac{E(H_i(S), H_i(T))}{\sqrt{|H_i(S)||H_i(T)|}} \geq \frac{d^*}{2}$, or we have, $E(H_i(S), H_i(T)) \geq \sqrt{|H_i(S)||H_i(T)|}\frac{d^*}{2} = \sqrt{a}|H_i(T)|\frac{d^*}{2}$. Hence by summing over the iterations 1 to $l_2 + 1$ we get, $E(S_{l_2+1}, T_{l_2+1}) \geq \sqrt{a}\sum_{i=1}^{l_2+1}|H_i(T)|\frac{d^*}{2} \geq \frac{d^*}{2}\sqrt{a}|T_{l_2+1}| = \frac{d^*}{2}\sqrt{|T_{l_2+1}||S_{l_2+1}|}$. Hence we have, $\frac{E(S'_{l_2+1}, T'_{l_2+1})}{\sqrt{|S'_{l_2+1}||T'_{l_2+1}|}} = \frac{E(S_{l_2+1}, T_{l_2+1})}{\sqrt{|S_{l_2+1}||T_{l_2+1}|}} \geq \frac{d^*}{2}$.

When case 3 occurs, we have again $\forall i = 1, 2, .., l'$, $\frac{E(H_i(S), H_i(T))}{\sqrt{|H_i(S)||H_i(T)|}} \geq \frac{d^*}{2}$. Now following the same analysis as in case 2, $\frac{E(S_{l'}, T_{l'})}{\sqrt{|S_{l'}||T_{l'}|}} \geq \frac{d^*}{2}$. Since $|S'_{l'}| = |S_{l'}|$ might be much larger than k_1, the analysis as in case 1 cannot guarantee a 2-approximation. Let $X_1 = |\bigcup_{i=1}^{l'} H_i(S)| = |S_{l'}|$ and $X_2 = \sum_{i=1}^{l'}|H_i(S)|$. Similarly $Y_1 = |\bigcup_{i=1}^{l'} H_i(T)| = |T_{l'}|$ and $Y_2 = \sum_{i=1}^{l'}|H_i(T)|$. We have $Y_2 = \frac{X_2}{a} \geq \frac{X_1}{a} \geq \frac{k_1}{a} = k_2$. Also we have, $\frac{E(X_1, Y_1)}{\sqrt{|X_2||Y_2|}} \geq \frac{d^*}{2}$. So $\frac{E(X_1, Y_1)}{\sqrt{|S'_{l'}|}} = \frac{E(X_1, Y_1)}{\sqrt{|X_1|}} \geq \frac{E(X_1, Y_1)}{\sqrt{|X_2|}} \geq \sqrt{Y_2}\frac{d^*}{2} \geq \sqrt{k_2}\frac{d^*}{2}$. We add arbitrary vertices to Y_1 to make its size equal to k_2. Hence, $\frac{E(X_1, Y_1)}{\sqrt{|S'_{l'}|}} \geq \sqrt{|T'_{l'}|}\frac{d^*}{2}$. Also $\frac{E(X_1, Y_1)}{\sqrt{|T'_{l'}|}} = \frac{E(X_1, Y_1)}{\sqrt{k_2}} \geq \frac{E(X_1, Y_1)}{\sqrt{|Y_2|}} \geq \sqrt{|X_2|}\frac{d^*}{2} \geq \sqrt{|S'_{l'}|}\frac{d^*}{2}$. Multiplying we get, $\frac{E(S'_{l'}, T'_{l'})^2}{\sqrt{|S'_{l'}||T'_{l'}|}} \geq \sqrt{|S'_{l'}||T'_{l'}|}\frac{d^{*2}}{4}$, or we have, $\frac{E(S'_{l'}, T'_{l'})}{\sqrt{|S'_{l'}||T'_{l'}|}} \geq \frac{d^*}{2}$. □

For a particular value of a, using parametric max-flow, algorithm *Densest At least-*k_1, k_2 requires time of a single flow computation within a constant factor. However there are $|V|^2$ possible choices of a. An algorithm that does not need to guess the value of a, like *Densest- Subgraph-Directed*, is needed to reduce the time complexity, which is still open. Here we note that a LP based solution for this problem can be designed in the line of LP based algorithm for *DalkS* (details omitted).

4 Densest at most k subgraph problem

Densest at most k subgraph problem (*DamkS*) tries to find a subgraph of highest density, whose size is at most k. Andersen et. al. [2] showed that an α approximation for *DamkS* implies a $\Theta(\alpha^2)$ approximation for the densest k subgraph problem. We prove that approximating *DamkS* is as hard as the *DkS* problem, within a constant factor. Precisely we prove the following theorem:

Theorem 7. *An α approximation algorithm for* DamkS *implies an 4α approximation algorithm for the densest k subgraph problem*

The proof can be found in an extended version [14].

5 Conclusion

In this paper, we have discussed different variations of the densest subgraph problems with and without size constraints. We have considered hardness issues related to these problems and have developed fast algorithms for them for both undirected and directed networks. All these problems can be generalized to weighted setting, with same time-complexity or sometimes with only a $\log |V|$ increase in running time. An interesting open question will be to design linear time algorithm with an approximation factor better than 2 for densest subgraph without any size constraint or to improve the approximation factor for *DalkS* problem. Obtaining faster algorithms for *densest at lease-k_1, k_2 subgraph* problem, or removing the requirement of guessing a in it or in the flow graph construction of maximum density directed subgraph will also be useful.

References

1. R. Andersen. Finding large and small dense subgraphs. *CoRR*, abs/cs/0702032, 2007.
2. R. Andersen and K. Chellapilla. Finding dense subgraphs with size bounds. In *WAW '09*, pages 25–36, 2009.
3. Y. Asahiro, R. Hassin, and K Iwama. Complexity of finding dense subgraphs. *Discrete Appl. Math.*, 121(1-3):15–26, 2002.
4. Y. Asahiro, K. Iwama, H. Tamaki, and T. Tokuyama. Greedily finding a dense subgraph. In *SWAT '96*, pages 136–148, 1996.
5. G. Buehrer and K. Chellapilla. A scalable pattern mining approach to web graph compression with communities. In *WSDM '08*, pages 95–106, 2008.
6. M Charikar. Greedy approximation algorithms for finding dense components in a graph. In *APPROX*, pages 84–95, 2000.
7. Y. Dourisboure, F. Geraci, and M. Pellegrini. Extraction and classification of dense communities in the web. In *WWW '07*, pages 461–470, 2007.
8. U. Feige, G. Kortsarz, and D. Peleg. The dense k-subgraph problem. *Algorithmica*, 29:410–421, 1997.
9. G. Gallo, M. D. Grigoriadis, and R. E. Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM J. Comput.*, 18(1):30–55, 1989.
10. D. Gibson, R. Kumar, and A. Tomkins. Discovering large dense subgraphs in massive graphs. In *VLDB '05*, pages 721–732, 2005.
11. A. V. Goldberg. Finding a maximum density subgraph. Technical report, 1984.
12. R. Kannan and V. Vinay. Analyzing the structure of large graphs. Technical report, 1999.
13. S. Khot. Ruling out ptas for graph min-bisection, dense k-subgraph, and bipartite clique. *SIAM J. Comput.*, 36(4):1025–1071, 2006.
14. S. Khuller and B. Saha. On finding dense subgraphs. *http://www.cs.umd.edu/∼samir/grant /ICALP09.pdf*, 2009.
15. J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.
16. E. Lawler. *Combinatorial optimization - networks and matroids*. Holt, Rinehart and Winston, New York, 1976.