Fast algorithms for Constrained Graph Density problems

Venkatesan Chakaravarthy[1], Neelima Gupta[2], Aditya Pancholi[2], and Sambuddha Roy[3]**

[1] IBM Research, Bangalore tcvenkat@in.ibm.com
[2] University of Delhi, Delhi ngupta@cs.du.ac.in, aditya.cs.du@gmail.com
[3] Amazon, Bangalore shombuddho@gmail.com

Abstract. We consider the question of finding communities in *large* social networks. In literature and practice, "communities" refer to a *well-connected* subgraph of the entire network. For instance, the notion of *graph density* has been considered as a reasonable measure of a community. Researchers have also looked at the *minimum degree* of a subgraph as a measure of the connectedness of the community.

Typically, a community is meaningful in the context of a social network if it is of somewhat significant size. Thus, earlier work has considered the densest graph problem subject to various *co-matroid constraints*. Most of these algorithms utilize an exact dense subgraph procedure as a subroutine; such a subroutine involves computing maximum flows or solving LPs. Consequently, they are rather inefficient when considered for massive graphs. For massive graphs, we are constrained to run in near-linear time, while producing subgraphs that provide reasonable approximations to the optimal solutions.

Our current work presents efficient greedy algorithms for the problem of graph density subject to an even more general class of constraints called *upward-monotone* constraints (these subsume co-matroid constraints). This generalizes and extends earlier work significantly. For instance, we are thereby able to present near-linear time 3-factor approximation algorithms for density subject to co-matroid constraints; we are also able to obtain 2-factor LP-based algorithms for density subject to 2 co-matroid constraints.

Our algorithms heavily utilize the *core decomposition* of a graph.

1 Introduction

Given an undirected graph $G = (V, E)$, the density $d(S)$ of a subgraph on vertex set S is defined as the quantity $\frac{|E(S)|}{|S|}$, where $E(S)$ is the set of edges in the subgraph induced by the vertex set S. The densest subgraph problem is to find the subgraph S of G that maximizes the density.

The concept of graph density is ubiquitous, more so in the context of social networks. In the context of social networks, the problem is to detect *communities*:

** Work done while at IBM Research, India.

collections of individuals who are relatively well connected as compared to other parts of the social network graph.

The notion of graph density has been fruitfully applied to finding communities in the social network graph (or even web graphs, gene annotation graphs [17], problems related to the formation of most effective teams [8], etc.). Also, note that graph density appears naturally in the study of threshold phenomena in random graphs, see [1].

Motivated by applications in social networks, the graph density problem and its variants have been well studied. Goldberg [9] (also see [14]) proved that the densest subgraph problem can be solved optimally in polynomial time: he showed this via a reduction to a series of max-flow computations. Later, others [5, 12] have given new proofs for the above result, motivated by considerations to extend the result to some generalizations and variants.

Researchers also looked at the graph density problem subject to *size constraints*. In the k-densest subgraph problem, the objective is to find the densest subgraph induced over precisely k vertices. The problem is notoriously hard: while it was shown to be NP-hard by [7], Khot [11] shows that that there does not exist any PTAS for the k-densest subgraph problem under a reasonable complexity assumption. In the direction of upper bounds, [7] provide an algorithm with an approximation factor of $\mathcal{O}(n^{\theta})$ where $\theta < 1/3$. More recently, [3] give an algorithm for the k-densest subgraph with an approximation factor of $\mathcal{O}(n^{1/4})$.

Andersen and Chellapilla [2] considered two variants of the k-densest subgraph problem (they show both of the variants to be NP-hard). In one variant, the integer k provided along with the input specifies an *upper bound* on the size of the dense subgraph. They prove that this problem is almost as hard as the k-densest subgraph problem. The other variant they consider is where k specifies a *lower bound* on the size of the dense subgraph. Thus, here, the input includes an integer k and the goal is to find the densest subgraph S subject to the constraint $|S| \geqslant k$. This corresponds to finding *sufficiently large* dense subgraphs in social networks. Khuller and Saha [12] give two alternative algorithms for this problem: one of them is a greedy procedure that involves a dense subgraph subroutine, while the other is LP-based. Both the algorithms have 2-factor guarantees. However, these algorithms do not scale to *large* graphs such as web graphs because of the prohibitive runtime for the (exact) dense subgraph routine (or the LP solving routine). In this context, Andersen and Chellapilla [2] give an efficient approximation for this problem. They show that the problem may be approximated within a factor of 3 in linear time. Their algorithm is based on the *core decomposition* of a graph (see [13]).

Gajewar and Sarma [8] consider a further generalization, motivated by aspects of team formation (also see [16]). Call this the *team-formation* variant of graph density. Here, the input includes a partition of the vertex set into $U_1, U_2, \cdots, U_t$, and non-negative integers $r_1, r_2, \cdots, r_t$. The goal is to find the densest subgraph S subject to the constraint that for all $1 \leqslant i \leqslant t$, $|S \cap U_i| \geqslant r_i$. They gave a 3-approximation algorithm by extending the greedy procedure of

Khuller and Saha [12] (and thus involves an exact dense subgraph routine and has a prohibitive runtime).

Chakaravarthy et al. [4] vastly generalize the Gajewar & Sarma setting to that of *co-matroid constraints* (the precise definition of co-matroid constraints appears in Section 4). Co-matroid constraints capture both the cardinality constraints considered in [2, 12] as well as the partition constraints considered in [8]. Chakaravarthy et al. provide a 2-factor approximation algorithm for the densest subgraph problem subject to arbitrary co-matroid constraints thereby improving the approximation factor of [8]. Their algorithm heavily utilizes an exact dense subgraph routine.

The work in [4] also considers another class of constraints called *dependency constraints* (this class of constraints generalize *subset constraints* as considered by [17]). They show that the problem of finding the densest subgraph subject to such constraints is in polynomial time - this involves finding solutions to LPs.

Summarizing the above discussion, we see that efficient linear time algorithms exist for the vanilla version of graph density and the variant where the size of the solution subgraph is lower bounded by k. For other variants, such as the team-formation variant, or the more general co-matroid constraint variant, the existing algorithms depend on an *exact graph density* routine. In turn, the exact graph density routines typically depend on certain *maximum flow* formulations on derived directed graphs (see, for instance [12]). To date, we do not know linear time (or even nearly linear time) algorithms for these specific maximum flow formulations; this is why the exact graph density routines (and consequently, the algorithms for the co-matroid constrained versions of graph density), to have prohibitive runtimes.

In this context, it ought to be mentioned that various recent papers (e.g. [19, 10], see also [6]) exhibit an exciting line of work that provide *near-linear time* algorithms for *nearly* maximum flow; however, these results apply to *undirected graphs*.

To summarise, the principal motivation of the current work is the following: to devise efficient (i.e. *near linear*) algorithms for these general variants of graph density.

We may also ask a stronger question: can we replace the *exact* densest subgraph routine with an efficient *approximate* routine and still preserve the overall approximation guarantee?

Two related objectives: As an aside, let us remark on a related objective that has been looked at previously in the context of community detection. Given a subgraph H of G, note that *twice* the density of H is precisely the *average* degree of the subgraph H. Thus the densest subgraph problem may also be restated as the problem of finding the subgraph *maximizing* the *average* degree. Given this, a related objective may also be of finding the subgraph *maximizing* the *minimum* degree.

Thus, while the densest subgraph problem corresponds to the problem of finding a subgraph where the degrees of the vertices are large, *on average*, the

problem of maximizing the minimum degree corresponds to finding a subgraph where the degrees of the vertices are large *in the worst-case*.

It turns out that the worst-case problem is significantly simpler than the average case question; in fact the *core decomposition* procedure [13] yields a subgraph with the maximum minimum degree - this was first proven by Matula & Beck [15].

Since a subgraph with the maximum minimum degree is also a reasonable notion of *community*, various authors have studied this problem under different constraints. An *upward-monotone* constraint is such that if S is feasible, then any superset $S' \supseteq S$ is also feasible. Sozio & Gionis [20] show that the problem of maximizing the minimum degree subject to *any* upward-monotone constraint is solvable in linear time.

2 Main Contributions

The principal message of our work is conceptual. Thus far in literature, most of the work on graph density has focused on specific constraints such as *co-matroid* constraints, *dependency* constraints, etc. We explore the nature of constraints subject to which we may expect to have reasonable approximations for graph density. In the realm of *upward-monotone* constraints $\mathcal{C}$, the answer that we obtain has the following pleasing form. We are able to demonstrate a reduction from the problem of constrained graph density to that of a certain *Extension problem* for the constraints $\mathcal{C}$ (denoted as $\mathbf{Extend}_{\mathcal{C}}$; see Figure 3):

Theorem 1. *Suppose for some class of upward-monotone constraints $\mathcal{C}$, the problem $\mathbf{Extend}_{\mathcal{C}}(S)$ admits a f-factor approximation. Then there is a $(f+1)$-factor approximation algorithm for the densest subgraph problem subject to constraints $\mathcal{C}$.*

*Similarly, suppose the problem $\mathbf{Extend}_{\mathcal{C}}(S)$ admits an **efficient** f-factor approximation. Then there is an **efficient** $(f+2)$-factor approximation algorithm for the densest subgraph problem subject to constraints $\mathcal{C}$.*

The content of the result above is that we are able to translate an approximation factor for the $\mathbf{Extend}_{\mathcal{C}}$ problem to (almost) the same approximation factor for the constrained density problem.

Furthermore, this holds even when we are considering *efficient* (i.e. near-linear time) algorithms; we believe that this is the novel technical contribution of this paper.

This allows us to approach the domain of *web-scale* graphs in the context of such *constrained* graph density problems. We prove Theorem 1 in Section 5.

We may instantiate the above general result with various constraints as in the corollaries below.

Corollary 1. *There is a linear time algorithm for the densest subgraph problem subject to co-matroid constraints that achieves a 3-factor approximation guarantee.*

This is because the problem $\mathbf{Extend}_{\mathcal{C}}(S)$ in this case has an *optimal* (i.e. 1-factor) linear-time algorithm [4].

To exhibit the generality of our result, consider the following (perhaps contrived) problem. Given a graph, we are given the task of finding the densest subgraph that also forms a *vertex cover*. We can prove:

Corollary 2. *There is a* 3*-factor* LP*-based algorithm and a* 4*-factor linear time algorithm for this problem.*

3 Other Results: Combinations of constraints

We are also able to show certain results for *combinations* of constraints.

Two Co-matroid Constraints

- Consider the densest subgraph problem subject to *two* co-matroid constraints. Hithertofore, no approximation algorithms were given for this. We prove that this densest subgraph variant admits an LP-based 2-factor approximation (here, the $\mathbf{Extend}_{\mathcal{C}}$ routine corresponds to matroid intersection).
- For the specific case of the 2 co-matroid constraints corresponding to a bipartite matching, the densest subgraph problem admits a 3-factor *linear time* algorithm.

We are also able to show results for the minimum degree objective under a *combination* of upward-monotone and dependency constraints.

Co-matroid + Dependency Constraint

In the case of the *max min degree* objective, we are able to show the optimal result:

Theorem 2. *There is an efficient* optimal *algorithm for the problem of maximizing the minimum degree of a subgraph subject to any upward-monotone constraint and dependency constraints.*

This generalizes the results of Sozio & Gionis [20]; due to space limitations, this is proven in Appendix A.1.

For the densest subgraph objective (i.e. max average degree), we are only able to derive partial results. We are able to provide constant factor guarantees for the case when the (i) co-matroid constraint corresponds to a general cardinality constraint (i.e. S is feasible if $|S \cap A| \geqslant k$ for a specific subset A) and (ii) the dependency constraint corresponds to a *tree*. Note however, this already generalizes the *subset constraint* case considered by Khuller & Saha [17]. For this case, we are able to show (proof sketch in Appendix A.5):

Theorem 3. *There exists a* LP*-based algorithm that achieves a* 2*-factor approximation for the densest subgraph problem subject to the following constraints: (i) a cardinality co-matroid, i.e. feasible S's satisfy: $|S \cap A| \geqslant k$ for some set A, and number k, and (ii) a dependency constraint corresponding to a tree.*

Knapsack Constraints

For a knapsack constraint, the densest subgraph problem admits a 3-factor greedy linear time algorithm (the corresponding $\mathbf{Extend}_{\mathcal{C}}$ routine admits an efficient 1-factor approximation).

4 Preliminaries

Given a graph G, we will denote the set of vertices of the graph as $V(G)$ and the set of edges as $E(G)$. If the graph G is clear from the context, then we may refer to the set of vertices (or edges) as V (or E). Given a graph G, the minimum degree in G is denoted by $\text{MinDeg}(G)$.

Supermodular functions:

A set function $f : 2^U \to \mathbb{R}^+$ over a universe U is called *supermodular* if the following holds for any two sets $A, B \subseteq U$: $f(A) + f(B) \leqslant f(A \cup B) + f(A \cap B)$.

In this paper, we will use the following equivalent definition of supermodularity (the "increasing marginal returns" perspective). Given disjoint sets A, B and C: $f(A + C) - f(A) \leqslant f(A + B + C) - f(A + B)$. The main fact that we will use in this paper is that: given a graph $G = (V, E)$, the set function $E(S)$ consisting of the edges within vertices of the set S is supermodular.

Core Decomposition:

Our algorithms heavily use the concept of a *core*. Given a graph G and a degree parameter $d \in \mathbb{N}$, the core $C(G, d)$ of the graph G is the *maximal* subgraph H of G such that the minimum degree of a vertex $v \in V(H)$ in the induced subgraph H is at least d. It is easy to notice that there is a *unique* such maximal subgraph, so that the notion of a core is well defined. We will utilize the actual core decomposition as a subroutine in our algorithm for Theorem 1, and the core decomposition algorithm is presented there in Figure 2.

The core decomposition of a graph is obtained via the *minimum degree* (MD) ordering of the graph. The MD ordering is an ordering of the vertices defined as follows (see [15]). The first vertex of the ordering is $v_1 = \arg\min_{v \in V(G)}\{\deg(v)\}$. Consider the graph $G_1 = G \setminus \{v_1\}$ obtained by deleting the vertex v_1 from $G_0 = G$. In the i^{th} iteration (for $i = 0, 1, \cdots, |V(G)| - 1$), we consider the graph G_i and set $v_{i+1} = \arg\min_{v \in V(G_i)}\{\deg_{G_i}(v)\}$. and $G_{i+1} = G_i \setminus \{v_{i+1}\}$. The MD ordering of the graph G is denoted as $\text{MD}(G) = \{v_1, v_2, \cdots, v_n\}$. Note that the sequence of degrees $\deg_{G_i}(v_{i+1})$ is not necessarily a monotone increasing sequence.

It is well known that every core is *some suffix* of the MD ordering (see [15, 13, 5] for more facts about cores).

Upward-Monotone Constraints:

An *upward-monotone* constraint is one such that if S is feasible, every superset of S is also feasible.

Co-matroid constraints (defined next) are a prominent subclass of upward-monotone constraints.

Co-Matroid Constraints:
A matroid is a pair $\mathcal{M} = (U, \mathcal{I})$ where $\mathcal{I} \subseteq 2^U$, and

1. (Hereditary Property) $\forall B \in \mathcal{I}, A \subset B \implies A \in \mathcal{I}$.
2. (Extension Property) $\forall A, B \in \mathcal{I} : |A| < |B| \implies \exists x \in B \setminus A : A + x \in \mathcal{I}$

Typically the sets in $\mathcal{I}$ are called *independent sets*, in keeping with the notions of linear algebra (see the excellent text by Schrijver [18] for details). A *co-matroid* constraint is defined as follows. Given a matroid $\mathcal{M} = (U, \mathcal{I})$, a set S is considered feasible iff the complement of S is *independent* in $\mathcal{I}$. Two commonly encountered matroids are cardinality matroids and partition matroids.

Dependency constraints:
Owing to space limitations, our results on Dependency constraints appear in Appendix A.1. We formally define such constraints in the Appendix too.

Structure of the density problem
It may be observed that most algorithms [4, 12, 2] for graph density (subject to upward-monotone constraints) work in *two* phases. Our main algorithm (see Figure 1) also follows a similar two-phase paradigm. The first phase of the algorithm may be viewed as aiming to construct the unconstrained densest subgraph, and the second phase performs *augmentations* to the candidate solutions from the first phase in order to obtain feasible solutions for the constrained problem. This raises the question: is it always true that there is a constrained optimum that *contains* the unconstrained optimum? The (affirmative) answer is given by the following lemma; this is new to the best of our knowledge.

Lemma 1. *Consider the upward-monotone constrained density problem for graph $G = (V, E)$ and let H denote an unconstrained optimum (i.e. **a** subgraph with the highest density). Then there exists a constrained optimum solution C such that $H \subseteq C$.*

This lemma may be viewed as justification for the two-phase approaches alluded to above. This may also be interpreted as a reason as to why density problems subject to upward-monotone constraints seem easier than when subject to upper bound constraints (like the k-densest subgraph problem). For space limitations, we defer the proof to Appendix A.2.

5 Proof of Theorem 1

The algorithm consists of two principal routines: **ConstructCandidates**(G) and **Extend**$_{\mathcal{C}}(S)$. Interesting, the **ConstructCandidates** routine does not require the specific set of constraints $\mathcal{C}$. The **ConstructCandidates** routine outputs a list of candidate subsets $D_1, D_2, \cdots, D_r$. In fact, this is a **chain**: that is, $D_1 \subseteq D_2 \subseteq \cdots D_r$. Although the generic algorithm as presented in Figure 1

Density(G, $\mathcal{C}$)
Input: a graph G, an upward-monotone constraint $\mathcal{C}$
Output: a (dense) subgraph H.

$D_1, \cdots, D_r \leftarrow$ **ConstructCandidates**(G)
for $i \in \{1, 2, \cdots, r\}$ **do**
  $D'_i \leftarrow$ **Extend**$_{\mathcal{C}}(D_i)$
end for
$H \leftarrow$ the subgraph among D'_i (for $i = 1, \cdots, r$) with the highest density
Output H

Fig. 1. Generic Algorithm

does not explicitly utilize this fact, it turns out, that given this sequence one only need consider the *first feasible* solution in this chain along with the **Extend**$_{\mathcal{C}}$-ed solutions.

We also assume that there is a *near-linear* time algorithm for the **Extend**$_{\mathcal{C}}$ routine, that outputs a f-factor approximation.

We may implement the **ConstructCandidates** routine by repeatedly solving LP's; this is the approach adopted by earlier papers (see for instance [4]). Such an approach is possible for the current scenario with general upward-monotone constraints and would result in a $(f+1)$-factor approximation. However, such an algorithm would be costly, involving several LP-solving/max-flow steps. In the current work, we save over the costly LP-solving steps, incurring only a slightly worse $(f+2)$-factor, achieving this in *near linear* time.

Our efficient algorithms use the *core decomposition* routine (presented in Figure 2). Given this, we prove only the latter half (the part dealing with *efficient* algorithms) of Theorem 1.

Proof. Let H^* be the optimal densest subgraph (of density d^*) subject to the constraints $\mathcal{C}$.

We will prove that some subgraph among D'_i (for $i = 1, \cdots, r$) has the correct density, i.e. has density $\geqslant d^*/(f+2)$.

Note that, by construction, the sets in the sequence $D_1 \subseteq D_2 \subseteq \cdots \subseteq D_r$ satisfy the following relation: $\text{MinDeg}(D_1) > \text{MinDeg}(D_2) > \cdots > \text{MinDeg}(D_r)$. We may also note that the maximum minimum degree over a subgraph of G is attained by D_1.

In fact, we will be able to prove a sharper statement. Given the ordered sequence $D_1 \subseteq D_2 \subseteq \cdots D_r$, let L be the least index $(1 \leqslant L \leqslant r)$ such that D_L is feasible. We will then prove that *some* subgraph among D'_i (for $i = 1, \cdots, (L-1)$) and D_L has the correct density (i.e. density $\geqslant d^*/(f+2)$).

We will consider the subgraph D_ℓ such that the following holds (the *boundary condition*): The minimum degree in the subgraph D_i is at least $2d^*/(f+2)$ for $i = 1 \cdots \ell$ and the minimum degree in the subgraph $D_{\ell+1}$ is $< 2d^*/(f+2)$.

ConstructCandidates$(G) \leftarrow$ **Core**(G)
Input: a graph G.
Output: subsets $D_1, D_2, \cdots, D_r$ of vertices of G.
$i \leftarrow 1$
$K_i \leftarrow G$
while $K_i \neq \emptyset$ **do**
  $H \leftarrow K_i$
  while MinDeg$(H) \leqslant$MinDeg(K_i) **do**
    $v \leftarrow v_{\min}(H)$
    $H \leftarrow H \setminus \{v\}$
  end while
  $i \leftarrow i + 1$
  $K_i \leftarrow H$
end while
Let the sets constructed be $K_1 \supseteq \cdots \supseteq K_r$
Rename the sets as $D_i = K_{r+1-i}$, so that $D_1 \subseteq \cdots \subseteq D_r$.

Fig. 2. Routine **Core**(G)

Extend$_{\mathcal{C}}(S)$
Input: a set S, and a collection of *upward-monotone* constraints $\mathcal{C}$.
Output: a set T such that $S \subseteq T$ and T is *minimal feasible* for the constraints $\mathcal{C}$.

Fig. 3. Routine **Extend**$_{\mathcal{C}}(S)$

Why should such an ℓ even exist? Given that there is a subgraph in G of density d^*, the densest subgraph (without constraints) has density at least d^*. Let H denote the *unconstrained* densest subgraph of density at least d^*. It is easy to check that MinDeg$(H) \geqslant d^*$. Thus, G has a subgraph of minimum degree at least d^*; thus the maximum minimum degree of G (that is attained by the subgraph D_1) is at least $d^* > 2d^*/(f+2)$.

If $\ell = L$ (i.e. if D_ℓ is feasible), then note that MinDeg$(D_\ell) \geqslant 2d^*/(f+2)$; thus, the density of the subgraph D_ℓ is $\geqslant d^*/(f+2)$.

Thus, suppose that $\ell < L$, so that D_ℓ is *not* feasible. In this case, we will prove that the set D'_ℓ has the *correct* density i.e. $d(D'_\ell) \geqslant d^*/(f+2)$.

To this end, we will prove two lemmas about the set D_ℓ:

Lemma 2. *The following holds:*

$$|E(D_\ell)| - |E(D_\ell \cap H^*)| \geqslant \frac{d^*}{(f+2)}(|D_\ell| - |D_\ell \cap H^*|)$$

Proof. Let X denote the vertices in $D_\ell \setminus H^*$. Note that $|X| = |D_\ell| - |D_\ell \cap H^*|$. Also, since $X \subseteq D_\ell$, every vertex in X has degree at least $2d^*/(f+2)$, by definition of the subset D_ℓ.

The sum $\sum_{v\in X} \mathrm{Deg}_{D_\ell}(v)$ counts each edge $e \in E(D_\ell \setminus H^*)$ *twice* and every edge $e \in \delta(D_\ell \setminus H^*)$ once. Thus, we have that $\sum_{v\in X} \mathrm{Deg}_{D_\ell}(v) = 2|E(D_\ell \setminus H^*)| + |\delta(D_\ell \setminus H^*)|$. Hence, it also holds that

$$2(|E(D_\ell)| - |E(D_\ell \cap H^*)|) = 2|E(D_\ell \setminus H^*)| + 2|\delta(D_\ell \setminus H^*)| \geqslant \sum_{v\in X} \mathrm{Deg}_{D_\ell}(v) \geqslant \frac{2d^*}{(f+2)}|X|.$$

This proves the statement of the lemma.

The next lemma attempts to lowerbound the value of $|E(D_\ell \cup H^*)|$.

Lemma 3. *The following holds:*

$$|E(D_\ell \cup H^*)| - |E(D_\ell)| < \frac{2d^*}{(f+2)}(|H^*| - |D_\ell \cap H^*|)$$

Proof. Consider the set $D_\ell \cup H^* \setminus D_\ell$. Assume that there are k vertices in this set; thus, $k = |H^*| - |D_\ell \cap H^*|$. We will prove that these vertices may be ordered as $\{h_1, h_2, \cdots, h_k\}$ such that for any i, the number of edges between h_{i+1} and $D_\ell \cup \{h_1, h_2, \cdots, h_i\}$ is *less than* $\frac{2d^*}{(f+2)}$. Note that this would prove the lemma; the quantity $|E(D_\ell \cup H^*)| - |E(D_\ell)|$ may be decomposed as $\sum_{i=1}^{k} |E(h_i, D_\ell \cup \{h_1, h_2, \cdots, h_{i-1}\})|$.

To this end, we will consider the *reverse of* the MD ordering of the vertices of $G = \{v_1, v_2, \cdots, v_n\}$. Thus v_n is the vertex of minimum degree in G. According to this notation let $D_\ell = \{v_1, v_2, \cdots, v_s\}$ for some s; also D_ℓ is the *core* corresponding to some degree $d \geqslant 2d^*/(f+2)$.

Also by definition of D_ℓ, for any set $\{v_1, v_2, \cdots, v_i\}$ (for $i > s$), it holds that the degree of v_i in the set is $< 2d^*/(f+2)$. Rephrased, this means that the number of edges between v_i and $\{v_1, v_2, \cdots, v_{i-1}\}$ is $< 2d^*/(f+2)$.

But now, the ordering of the h_i's is clear: it corresponds to the order in which the vertices h_i appear in the ordering $\{v_1, v_2, \cdots, v_n\}$. If a specific h_{i+1} is v_t in this ordering, then the number of edges between h_{i+1} and $D_\ell \cup \{h_1, h_2, \cdots, h_i\}$ is at most the number of edges between v_t and $\{v_1, v_2, \cdots, v_{t-1}\}$. However by the argument above, this is less than $2d^*/(f+2)$.

This completes the proof of this lemma.

Thus we may claim the following:

Lemma 4.

$$|E(D_\ell \cap H^*)| \geqslant \frac{2d^*}{(f+2)}|D_\ell \cap H^*| + \frac{fd^*}{(f+2)}|H^*|$$

Proof. Consider the following statement that follows from supermodularity of the $E(\cdot)$ function (using the *increasing marginal returns* perspective):

$$|E(H^*)| - |E(D_\ell \cap H^*)| \leqslant |E(D_\ell \cup H^*)| - |E(D_\ell)|$$

Thus, Lemma 3 implies that

$$|E(H^*)| - |E(D_\ell \cap H^*)| \leqslant \frac{2d^*}{(f+2)}(|H^*| - |D_\ell \cap H^*|)$$

Now, using that $E(H^*) = d^*|H^*|$ (since H^* is the densest constrained subgraph), and simplifying, we get that

$$|E(D_\ell \cap H^*)| \geqslant \frac{2d^*}{(f+2)}|D_\ell \cap H^*| + \frac{fd^*}{(f+2)}|H^*|$$

as required.

Now adding the statements of Lemma 4 and Lemma 2, we get that

$$|E(D_\ell)| \geqslant \frac{d^*}{(f+2)}(|D_\ell \cap H^*| + |D_\ell| + f|H^*|)$$

Given that D_ℓ is *infeasible*, the $\mathbf{Extend}_{\mathcal{C}}$ subroutine in the algorithm augments the set D_ℓ with some number of vertices in order to make it feasible (and the feasible solution is denoted as D'_ℓ).

We note that the number of extra vertices needed by the $\mathbf{Extend}_{\mathcal{C}}$ routine is upper bounded by $f|H^*|$ - this follows because, by assumption, we are given a f-factor approximation for the problem $\mathbf{Extend}_{\mathcal{C}}$ for the constraints $\mathcal{C}$. Thus, this means that the number of vertices in the subgraph D'_ℓ is at most $|D_\ell| + f|H^*|$. The above inequality then implies that (after dropping the term corresponding to $\frac{d^*}{(f+2)}|D_\ell \cap H^*|$):

$$|E(D'_\ell)| \geqslant \frac{d^*}{(f+2)}(|D_\ell| + f|H^*|) \geqslant \frac{d^*}{(f+2)}|D'_\ell|$$

so that the augmented set D'_ℓ has density at least $d^*/(f+2)$.

This concludes the proof of Theorem 1.

Runtime Analysis:

Deferred to Appendix A.3.

Remarks:

It is interesting to compare the current result with the earlier work by Andersen & Chellapilla [2]. They showed a greedy 3-factor approximation algorithm for problem of finding a dense subgraph subject to a *cardinality* constraint $|S| \geqslant k$. Their algorithm in a nutshell is this: consider the core decomposition of the graph; each core is a candidate solution. Look only at the cores that are *feasible* (i.e. has size $\geqslant k$), and output the *densest* of these subgraphs as the final solution.

However, when we consider general upward-monotone constraints (in fact, even constraints as simple as $|S \cap A| \geqslant k$), we are unable to claim such a performance guarantee by just restricting ourselves to looking *only* at the feasible solutions from the core decomposition.

In the general scenario of *upward-monotone constraints*, it appears that it is essential that we consider the cores in the **reverse** order, i.e. consider the infeasible cores and augment them to render them feasible.

References

1. Noga Alon and Joel H. Spencer. *The Probabilistic Method*. Wiley, New York, 1992.
2. Reid Andersen and Kumar Chellapilla. Finding dense subgraphs with size bounds. In *WAW*, pages 25–37, 2009.
3. Aditya Bhaskara, Moses Charikar, Eden Chlamtac, Uriel Feige, and Aravindan Vijayaraghavan. Detecting high log-densities: an $o(n^{1/4})$ approximation for densest k-subgraph. In *STOC*, pages 201–210, 2010.
4. Venkatesan T. Chakaravarthy, Natwar Modani, Sivaramakrishnan R. Natarajan, Sambuddha Roy, and Yogish Sabharwal. Density functions subject to a co-matroid constraint. In *FSTTCS*, pages 236–248, 2012.
5. Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. In *APPROX*, pages 84–95, 2000.
6. David Eisenstat and Philip N. Klein. Linear-time algorithms for max flow and multiple-source shortest paths in unit-weight planar graphs. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 735–744, 2013.
7. Uriel Feige, Guy Kortsarz, and David Peleg. The dense k-subgraph problem. *Algorithmica*, 29:2001, 1999.
8. Amita Gajewar and Atish Das Sarma. Multi-skill collaborative teams based on densest subgraphs. In *SDM*, pages 165–176, 2012.
9. A. V. Goldberg. Finding a maximum density subgraph. Technical report, UC Berkeley, 1984.
10. Jonathan A. Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 217–226, 2014.
11. Subhash Khot. Ruling Out PTAS for Graph Min-Bisection, Dense k-Subgraph, and Bipartite Clique. *SIAM J. Comput.*, 36(4):1025–1071, 2006.
12. Samir Khuller and Barna Saha. On finding dense subgraphs. In *ICALP*, 2009.
13. Guy Kortsarz and David Peleg. Generating sparse 2-spanners. *J. Algorithms*, 17(2):222–236, September 1994.
14. E. Lawler. *Combinatorial optimization - networks and matroids*. Holt, Rinehart and Winston, New York, 1976.
15. David W. Matula and Leland L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *J. ACM*, 30(3):417–427, July 1983.
16. Syama Sundar Rangapuram, Thomas Bühler, and Matthias Hein. Towards realistic team formation in social networks based on densest subgraphs. In *Proceedings of the* 22^{nd} *International Conference on World Wide Web*, WWW '13, pages 1077–1088, 2013.
17. Barna Saha, Allison Hoch, Samir Khuller, Louiqa Raschid, and Xiao-Ning Zhang. Dense subgraphs with restrictions and applications to gene annotation graphs. In *RECOMB*, pages 456–472, 2010.
18. A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer, 2003.
19. Jonah Sherman. Nearly maximum flows in nearly linear time. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 263–269, 2013.
20. Mauro Sozio and Aristides Gionis. The community-search problem and how to plan a successful cocktail party. In *KDD*, pages 939–948, 2010.

A Appendix

A.1 Proof of Theorem 2

We proceed by defining *dependency constraints.*

Definition 1. *(Dependency Constraints) In this class of constraints, along with the original graph G, we are also given an auxiliary digraph called the dependency graph $D = (V, \boldsymbol{A})$ defined over the same set of vertices as G. A feasible solution S has to satisfy the following property: if $a \in S$, then every vertex of the digraph D* reachable *from a also has to belong to S. Alternatively, $a \in S$ and $(a, b) \in \boldsymbol{A}$ implies that $b \in S$. The goal is to find the densest subset S subject to these dependency constraints. As noted in [4] dependency constraints generalize subset constraints. The motivation for such constraints comes from certain considerations in social networks, where we are to find the densest subgraph but with the restriction that in the solution subgraph all the members of a sub-community (say, a family) are present or absent simultaneously. We note that dependency constraints are incomparable with co-matroid constraints. In fact dependency constraints are not even upward monotone: it is* not *true that if S is a feasible subset,* any *superset of S is feasible.*

We may now consider the problem of maximizing the minimum degree in a graph $G = (V, E)$ subject to *upward-monotone* constraints $\mathcal{C}$ *and* dependency constraints (specified by the dependency graph $D = (V, \boldsymbol{A})$).

We provide a simple greedy approach for the above problem. We proceed to make a few definitions that will be used in the foregoing.

Definition 2 (Reachable set $R(v)$). *Given a vertex $v \in V$ looked in the context of the dependency graph D, let $R(v)$ denote the set of vertices* reachable *from the vertex v in the (directed) graph D.*

It is easy to check that if S and T are feasible for the dependency constraints, so also is $S \cup T$. Consider a vertex $v \in S \cup T$. Suppose, wlog, that $v \in S$. This means that $R(v) \in S \subseteq S \cup T$. This proves that $S \cup T$ is feasible too.

The cardinal definition for our algorithm is:

Definition 3. *Given a dependency graph D and a graph H (where $V(H) \subseteq V(D)$), and a vertex $v \in V(H)$, let $\deg_H(v)$ denote the degree of vertex v in the graph H. Given a vertex $v \in V(H)$, the effective degree (denoted as $\widehat{deg}(v)$) wrt the dependency graph D is defined as*

$$\widehat{deg}(v) = \min_{v' \in R(v) \cap V(H)} \{\deg_H(v)\}$$

The algorithm is presented in Figure 4. We prove that the algorithm produces an optimal solution to our problem.

First, we observe that the dependency constraints are satisfied across every iteration. To see this, suppose that, at some iteration i, graph K_i is feasible

$i \leftarrow 0$
$K_i \leftarrow G$
while K_i is feasible wrt the upward-monotone constraint $\mathcal{C}$ **do**
  $\text{MinDeg}_i \leftarrow \min_{v \in V(K_i)} \{\widehat{deg}_i(v)\}$
  $L \leftarrow \{v \in K_i : \widehat{deg}_i(v) = \text{MinDeg}_i\}$
  $K_{i+1} \leftarrow K_i \setminus L$.
  $i \leftarrow i + 1$
end while
Output K_j with the maximum MinDeg_j for $0 \leq j \leq i - 1$.

Fig. 4. Algorithm for maximizing minimum degree

wrt the dependency constraints given by A. Note that by definition of effective degree, if $u \in R(v) \cap V(K_i)$ then $\widehat{deg}_i(v) \leqslant \widehat{deg}_i(u)$. Thus u cannot be deleted before v, preserving the dependency constraints in iteration $(i + 1)$.

Let $\mathcal{O}$ be the optimal solution, with minimum degree d^*. If A and B are optimal solutions with minimum degree d^*, then $A \cup B$ is feasible wrt the upward-monotone constraints $\mathcal{C}$ as well as the dependency constraints. Thus $A \cup B$ is also optimal.

In the foregoing, we will assume that $\mathcal{O}$ is the *largest* optimal solution with minimum degree d^*. Now, we can make the following claim:

Claim. For each i, one of the following cases occurs: (i) the optimal solution $\mathcal{O}$ contains K_i or (ii) K_i contains a subgraph with minimum degree $\geqslant d^*$.

Proof. We will prove the statement via induction. Base Case: At the beginning, $\mathcal{O}$ is contained in $K_0 = G$. Thus, the claim is trivially true.

We will now assume that the claim is true for some K_l and prove the statement for K_{l+1}. If $K_l \subseteq \mathcal{O}$, then clearly, $K_{l+1} \subseteq K_l \subseteq \mathcal{O}$.

So let us assume that condition (ii) of the statement of the claim is true, i.e. K_l contains a subgraph (call this subgraph H) with minimum degree $\geqslant d^*$. If K_l is not optimal, then $\text{MinDeg}_l < d^*$. We want to prove that no vertex of H is deleted in this current iteration (where the vertices with minimum effective degree are removed from K_l). Since H is *contained* in K_l, for any vertex $u \in H$, it is true that $\widehat{deg}(u)$ in K_l is $\geqslant \widehat{deg}(u)$ in H. This latter $\widehat{deg}(u)$ in H is $\geqslant d^*$, so the effective degree (in K_l) of any $u \in H$ is $\geqslant d^*$. Thus the vertices of H may not be removed from K_l in this iteration, which implies that $H \subseteq K_{l+1}$. This implies that condition (ii) is true for K_{l+1} too, and we have proven the statement of the claim by induction.

Now consider the *greatest* i (call this $i^\star$) such that K_i contains a subgraph with minimum degree $\geqslant d^*$. By the above, $K_{i^\star+1}$ is contained in $\mathcal{O}$. If $\text{MinDeg}(K_{i^\star}) < d^*$, then the Claim implies that $K_{i^\star+1}$ contains a subgraph of minimum degree $\geqslant d^*$, so equals $\mathcal{O}$ (recall that we assumed that $\mathcal{O}$ is the *maximal* optimal subgraph). Otherwise, $\text{MinDeg}(K_{i^\star}) = d^*$ and we are done.

A.2 Proof of Lemma 1

Proof. Consider a constrained optimum solution C. We will prove that one of two things has to happen: either $H \subseteq C$ in which case we have nothing to prove, or that $H \cup C$ is also a constrained optimum solution. We proceed to prove this last statement.

Clearly, $H \cup C$ is feasible for the upward-monotone constraints. We need to show that the subgraph induced by $H \cup C$ has density at least that of C. Let the unconstrained density be d_u and the constrained maximum density be d_c (where $d_u \geqslant d_c$). Note that

$$\begin{aligned} |E(H \cup C)| + |E(H \cap C)| &\geqslant |E(H)| + |E(C)| \\ |E(H)| &= d_u|H| \\ |E(C)| &= d_c|C| \\ |E(H \cap C)| &\leqslant d_u|H \cap C| \end{aligned}$$

where the last inequality is because d_u is the maximum density. Thereby we get

$$\begin{aligned} |E(H \cup C)| &\geqslant d_u(|H| - |H \cap C|) + d_c|C| \\ &\geqslant d_c(|H| - |H \cap C|) + d_c|C| \\ &= d_c|H \cup C| \end{aligned}$$

Thus, $H \cup C$ has density at least as much as of C. Since C is the constrained optimum, so also is $H \cup C$. This proves a subclaim that there is a constrained optimum (namely $H \cup C$) that contains the unconstrained optimum. In order to prove that any constrained optimum contains H, we will have to work slightly more.

Note the equality cases of the inequalities in the lemma. Since C is the *optimum*, so also is the set $H \cup C$, and the inequalities above are all *tight*. But this implies that the supermodular inequality is tight, as also that $d_u = d_c$ or that $|H| = |H \cap C|$ (which means that H is already a subset of C). Thus, if $d_u \neq d_c$, then any constrained optimum *has to* contain the unconstrained optimum.

A.3 Runtime Analysis for Theorem 1

The linear runtime analysis for the algorithm in Figure 1 for Theorem 1 relies on the following. The core decomposition (Figure 2) of a graph $G = (V, E)$ may be obtained in time $\mathcal{O}(|V| + |E|)$ (see [13]).

Given the cores in sequence, the subroutine $\mathbf{Extend}_{\mathcal{C}}(S)$ is applied to all the cores. Superficially it may seem that the extension of the various cores may take time proportional to the number of cores and the time for each extension. However this entire process may be implemented in one pass over the vertices of the graph in the MD ordering. This is partially because the constraints $\mathcal{C}$ constitute co-matroid constraints and consequently, the $\mathbf{Extend}_{\mathcal{C}}$ algorithm is a trivial greedy algorithm. The main reason is that if $D_1 \subseteq D_2 \subseteq \cdots \subseteq D_r$ are the cores, the *extension* for D_i can be assumed to be a *superset* of the extension

for D_{i+1}. Also, in this single pass, one may maintain the counts of the edges and vertices included thereby outputting the densest subgraph among all the extended candidates. Details will appear in the full version of the paper.

A.4 Other Results

Most of the results in Section 3 are derived from Theorem 1, by instantiating the $\mathbf{Extend}_{\mathcal{C}}$ routine suitably.

- For $\mathcal{C}$ being 2 co-matroid constraints, the densest subgraph problem admits a 2-factor approximation. In this case the $\mathbf{Extend}_{\mathcal{C}}$ problem reduces to a *matroid intersection* problem (and is solvable in polynomial time). Here the $\mathbf{Extend}_{\mathcal{C}}()$ procedure is LP-based, and we use the LP procedure as the **ConstructCandidates** routine.
- For the specific case of the 2 co-matroid constraints corresponding to a bipartite matching, the densest subgraph problem admits a 4-factor *linear time* algorithm. The proof for the extension problem in this case utilizes the maximal matching algorithm for bipartite graphs, and will appear in the full version of the paper. Thus, we get a 2-factor for the $\mathbf{Extend}_{\mathcal{C}}$ procedure; we implement the **Core** procedure for the **ConstructCandidates** routine, thereby achieving a $(f+2) = 4$-factor approximation.

A.5 Proof of Theorem 3

In this problem, we are given a graph $G = (V, E)$, an upward *directed* tree T on the same set of vertices V (specifying the *dependency* constraint) and a set $A \subseteq V$ and an integer k (specifying the *co-matroid* constraint).

The algorithm follows the generic structure as outlined in Figure 1. In fact, the **ConstructCandidates**(G) routine is the same as that in [4]. We have to provide a suitable algorithm (and approximation guarantee) for the $\mathbf{Extend}_{\mathcal{C}}$ problem. Thus, the objective is to minimize the *size* of a feasible solution S, where feasibility implies that S is 'closed under successor' (i.e. according to the dependency graph T) and $|S \cap A| \geqslant k$. We say that a subset $S \subseteq V$ is closed under successor if $u \in S$ and $(u, v) \in E(T)$ imply that $v \in S$.

We employ a dynamic program to solve this problem *optimally*.

For each vertex $v \in V$, let $a(v)$ be the number of arcs coming into v (number of children of v), where $a(v) = 0$ if v is a leaf. For each $v \in V$ and $0 \leqslant i \leqslant a(v)$, let $T[v, i]$ denote the subtree induced by v, its first i children (left to right) and all their predecessors. Also let $d(v)$ denote the number of children of the node v. Note that $T[root(T), d(root(T))] = T$ and for each $v \in V$, $T[v, 0]$ is the subtree consisting of v alone.

A solution for a tree $T[root(T), d(root(T))]$ will be a non empty subset $S \subset V$ that is closed under successor and $k \leqslant |S \cap A|$. For each triplet $[v, i, q]$, we define $X[v, i, q]$, which is the maximum number of nodes from set A that can be included in the solution from the subtree $T[v, i]$ within the cost q. Note that this solution will be 'closed under successor'. In summary, we are trying to calculate minimum

$X[v, 0, q] = w(v)$ if $q \geqslant 1$ and 0 otherwise.
for $1 \leqslant i \leqslant d(v)$ and w is the i^{th} child of v
$X[v, i, q] = \max\{X[v, i-1, q], X[v, i-1, q'] + X[w, d(w), q - q']\}$
where $0 < q' < q$ (Note q and $q - q'$ are both strictly greater than 0)

Fig. 5. Recursion for the DP

value of q such that $X[root(T), d(root(T)), q] \geqslant k$. We know that $q \leqslant |V|$ (as in the worst case, we end up picking all nodes from T).

For simplicity, we define $w(v) = 1$ if $v \in A$ else 0. The value of $X[v, i, q]$ can be easily calculated using the formula in Figure 5.

Since $q \leqslant |V|$, the value of $X[root(T), d(root(T)), q]$ can be calculated in polynomial time.