Denser than the Densest Subgraph: Extracting Optimal Quasi-Cliques with Quality Guarantees

Charalampos E. Tsourakakis[1] Francesco Bonchi[2] Aristides Gionis[3]
Francesco Gullo[2] Maria A. Tsiarli[4]

[1]Carnegie Mellon University Pittsburgh PA, USA
[2]Yahoo! Research Barcelona, Spain
[3]Aalto University Espoo, Finland
[4]University of Pittsburgh Pittsburgh PA, USA

ABSTRACT

Finding dense subgraphs is an important graph-mining task with many applications. Given that the direct optimization of edge density is not meaningful, as even a single edge achieves maximum density, research has focused on optimizing alternative density functions. A very popular among such functions is the average degree, whose maximization leads to the well-known *densest-subgraph* notion. Surprisingly enough, however, densest subgraphs are typically large graphs, with small edge density and large diameter.

In this paper, we define a novel density function, which gives subgraphs of much higher quality than densest subgraphs: the graphs found by our method are compact, dense, and with smaller diameter. We show that the proposed function can be derived from a general framework, which includes other important density functions as subcases and for which we show interesting general theoretical properties. To optimize the proposed function we provide an additive approximation algorithm and a local-search heuristic. Both algorithms are very efficient and scale well to large graphs.

We evaluate our algorithms on real and synthetic datasets, and we also devise several application studies as variants of our original problem. When compared with the method that finds the subgraph of the largest average degree, our algorithms return denser subgraphs with smaller diameter. Finally, we discuss new interesting research directions that our problem leaves open.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: [Database Applications-Data Mining]

General Terms

Algorithms, Experimentation

Keywords

Graph mining, Dense subgraph, Quasi-clique

KDD'13, August 11–14, 2013, Chicago, Illinois, USA.

1. INTRODUCTION

Extracting dense subgraphs from large graphs is a key primitive in a variety of application domains [26]. In the Web graph, dense subgraphs may correspond to thematic groups or even spam link farms, as observed by Gibson et al. [18]. In biology, finding dense subgraphs can be used for discovering regulatory motifs in genomic DNA [16], and finding correlated genes [25]. In the financial domain, extracting dense subgraphs has been applied to, among others, finding price value motifs [12]. Other applications include graph compression [9], reachability and distance query indexing [21], and finding stories and events in micro-blogging streams [3].

Given a graph $G = (V, E)$ and a subset of vertices $S \subseteq V$, let $G[S] = (S, E[S])$ be the subgraph induced by S, and let $e[S]$ be the size of $E[S]$. The *edge density* of the set S is defined as $\delta(S) = e[S]/\binom{|S|}{2}$. Finding a dense subgraph of G would in principle require to find a set of vertices $S \subseteq V$ that maximizes $\delta(S)$. However, the direct maximization of δ is not a meaningful problem, as even a single edge achieves maximum density. Therefore, effort has been devoted to define alternative density functions whose maximization allows for extracting subgraphs having large δ and, at the same time, non-trivial size. Different choices of the density function lead to different variants of the dense-subgraph problem. Some variants can be solved in polynomial time, while others are **NP**-hard, or even inapproximable.

1.1 Background and related work

Cliques. A clique is a subset of vertices all connected to each other. The problem of finding whether there exists a clique of a given size in a graph is **NP**-complete. A *maximum* clique of a graph is a clique having maximum size and its size is called the graph's clique number. Håstad [20] shows that, unless $\mathbf{P} = \mathbf{NP}$, there cannot be any polynomial time algorithm that approximates the maximum clique within a factor better than $\mathcal{O}(n^{1-\epsilon})$, for any $\epsilon > 0$. Feige [13] proposes a polynomial-time algorithm that finds a clique of size $\mathcal{O}((\frac{\log n}{\log\log n})^2)$ whenever the graph has a clique of size $\mathcal{O}(\frac{n}{\log n^b})$ for any constant b. Based on this, an algorithm that approximates the maximum clique problem within a factor of $\mathcal{O}(n\frac{(\log\log n)^2}{\log n^3})$ is also defined. A *maximal* clique is a clique that is not a subset of any other clique. The Bron-Kerbosch algorithm [7] finds all maximal cliques in a graph.

Densest Subgraph. Let $G(V, E)$ be a graph, $|V| = n$, $|E| = m$. The average degree of a vertex set $S \subseteq V$ is de-

fined as $\frac{2e[S]}{|S|}$ The *densest-subgraph* problem is to find a set S that maximizes the average degree. The densest subgraph can be identified in polynomial time by solving a parametric maximum-flow problem [17, 19]. Charikar [10] shows that the greedy algorithm proposed by Asashiro et al. [6] produces a $\frac{1}{2}$-approximation of the densest subgraph in linear time.

In the classic definition of densest subgraph there is no size restriction of the output. When restrictions on the size $|S|$ are imposed, the problem becomes **NP**-hard. Specifically, the *DkS* problem of finding the densest subgraph of k vertices is known to be **NP**-hard [5]. For general k, Feige et al. [14] provide an approximation guarantee of $\mathcal{O}(n^{\alpha})$, where $\alpha < \frac{1}{3}$. The greedy algorithm by Asahiro et al. [6] gives instead an approximation factor of $\mathcal{O}(\frac{n}{k})$. Better approximation factors for specific values of k are provided by algorithms based on semidefinite programming [15]. From the perspective of (in)approximability, Khot [22] shows that there cannot exist any PTAS for the *DkS* problem under a reasonable complexity assumption. Arora et al. [4] propose a PTAS for the special case $k = \Omega(n)$ and $m = \Omega(n^2)$. Finally, two variants of the *DkS* problem are introduced by Andersen and Chellapilla [2]. The two problems ask for the set S that maximizes the average degree subject to $|S| \leq k$ (*DamkS*) and $|S| \geq k$ (*DalkS*), respectively. They provide constant factor approximation algorithms for DalkS and evidence that DamkS is hard. The latter was verified by [23].

Quasi-cliques. A set of vertices S is an α-quasi-clique if $e[S] \geq \alpha\binom{|S|}{2}$, i.e., if the edge density of the induced subgraph $G[S]$ exceeds a threshold parameter $\alpha \in (0, 1)$. Similarly to cliques, *maximum* quasi-cliques and *maximal* quasi-cliques [8] are quasi-cliques of maximum size and quasi-cliques not contained into any other quasi-clique, respectively. Abello et al. [1] propose an algorithm for finding a single maximal α-quasi-clique, while Uno [31] introduces an algorithm to enumerate all α-quasi-cliques.

1.2 Contributions

Extracting the densest subgraph (i.e., finding the subgraph that maximizes the average degree) is particularly attractive as it can be solved exactly in polynomial time or approximated within a factor of 2 in linear time. Indeed it is a popular choice in many applications. However, as we will see in detail next, maximizing the average degree tends to favor large subgraphs with not very large edge density δ. The prototypical dense graph is the *clique*, but, as discussed above, finding the largest clique is inapproximable. Also, the clique definition is too strict in practice, as not even a single edge can be missed from an otherwise dense subgraph. This observation leads to the definition of quasi-clique, whose underlying intuition is the following: assuming that each edge in a subgraph $G[S]$ exists with probability α, then the expected number of edges in $G[S]$ is $\alpha\binom{|S|}{2}$. Thus, the condition of the α-quasi-clique expresses the fact that the subgraph $G[S]$ has more edges than those expected by this binomial model.

Motivated by this definition, we turn the quasi-clique condition into an objective function. In particular, we define the density function $f_{\alpha}(S) = e[S] - \alpha\binom{|S|}{2}$, which expresses the *edge surplus* of a set S over the expected number of edges under the random-graph model. We consider the problem of finding the *best* α-quasi-clique, i.e., a set of vertices S that maximizes the function $f_{\alpha}(S)$. We refer to the subgraphs that maximize $f_{\alpha}(S)$ as optimal quasi-cliques. To the best of our knowledge, the problem of extracting optimal quasi-cliques from a graph has never been studied before. We show that optimal quasi-cliques are subgraphs of high quality, with edge density δ much larger than densest subgraphs and with smaller diameter. We also show that our novel density function comes indeed from a more general framework which subsumes other well-known density functions and has appreciable theoretical properties.

Table 1: Difference between densest subgraph and optimal quasi-clique on some popular graphs. $\delta = e[S]/\binom{|S|}{2}$ is the edge density of the extracted subgraph, D is the diameter, and $\tau = t[S]/\binom{|S|}{3}$ is the triangle density.

	densest subgraph				optimal quasi-clique			
	$\frac{\lvert S\rvert}{\lvert V\rvert}$	δ	D	τ	$\frac{\lvert S\rvert}{\lvert V\rvert}$	δ	D	τ
Dolphins	0.32	0.33	3	0.04	0.12	0.68	2	0.32
Football	1	0.09	4	0.03	0.10	0.73	2	0.34
Jazz	0.50	0.34	3	0.08	0.15	1	1	1
Celeg. N.	0.46	0.13	3	0.05	0.07	0.61	2	0.26

Our contributions are summarized as follows.

- We introduce a general framework for finding dense subgraphs, which subsumes popular density functions. We provide theoretical insights into our framework: showing that a large family of objectives are efficiently solvable while other subcases are **NP**-hard.
- As a special instance of our framework, we introduce the novel problem of extracting optimal quasi-cliques.
- We design two efficient algorithms for extracting optimal quasi-cliques. The first one is a greedy algorithm where the smallest-degree vertex is repeatedly removed from the graph, and achieves an additive approximation guarantee. The second algorithm is a heuristic based on the local-search paradigm.
- Motivated by real-world scenarios, we define interesting variants of our original problem definition: (i) finding the top-k optimal quasi-cliques, and (ii) finding optimal quasi-cliques that contain a given set of vertices.
- We extensively evaluate our algorithms and problem variants on numerous datasets, both synthetic and real, showing that they produce high-quality dense subgraphs, which clearly outperform densest subgraphs. We also present applications of our problem in data-mining and bioinformatics tasks, such as forming a successful team of domain experts and finding highly-correlated genes from a microarray dataset.

1.3 A preview of the results

Table 1 compares our optimal quasi-cliques with densest subgraphs on some popular graphs.[1] The results in the table clearly show that optimal quasi-cliques have much larger edge density than densest subgraphs, smaller diameters and larger triangle densities. Moreover, densest subgraphs are usually quite large-sized: in the graphs we report in Table 1, the densest subgraphs contain always more than the 30% of the vertices in the input graph. For instance, in the Football

[1] Densest subgraphs are extracted here with the exact Goldberg's algorithm [19]. As far as optimal quasi-cliques, we optimize f_{α} with $\alpha = \frac{1}{3}$ and use our local-search algorithm.

graph, the densest subgraph corresponds to the whole graph, with edge density < 0.1 and diameter 4, while the extracted optimal quasi-clique is a 12-vertex subgraph with edge density 0.73 and diameter 2. The Jazz graph contains a perfect clique of 30 vertices: our method finds this clique achieving perfect edge density, diameter, and triangle density scores. By contrast, the densest subgraph contains 100 vertices, and has edge density 0.34 and triangle density 0.08.

2. A GENERAL FRAMEWORK

Let $G = (V, E)$ be a graph, with $|V| = n$ and $|E| = m$. For a set of vertices $S \subseteq V$, let $e[S]$ be the number of edges in the subgraph induced by S. We define the following function.

DEFINITION 1 (EDGE-SURPLUS). *Let $S \subseteq V$ be a subset of the vertices of G, and let $\alpha > 0$ be a constant. Given any two strictly-increasing functions g and h, we define* edge-surplus f_α *as:*

$$f_\alpha(S) = \begin{cases} 0, & S = \emptyset, \\ g(e[S]) - \alpha h(|S|), & \textit{otherwise.} \end{cases}$$

The rationale behind the above definition is due to a counterbalancing of two contrasting terms: the first term $g(e[S])$ favors subgraphs abundant in edges, whereas the second term $-\alpha h(|S|)$ penalizes large subgraphs. Our framework for finding dense subgraphs is based on the following optimization problem.

PROBLEM 1 (optimal (g, h, α)-edge-surplus). *Given a graph $G = (V, E)$, a constant α, and a pair of strictly-increasing functions g, h, find a subset of vertices $S^* \subseteq V$ such that $f_\alpha(S^*) \geq f_\alpha(S)$, for all sets $S \subseteq V$. We refer to the set S^* as the optimal (g, h, α)-edge-surplus of the graph G.*

The edge-surplus definition subsumes numerous popular existing density measures.

- By setting $g(x) = h(x) = \log x$, $\alpha = 1$, the optimal (g, h, α)-edge-surplus problem becomes equivalent to maximizing $\log e[S] - \log |S| = \log \frac{e[S]}{|S|}$, which corresponds to the popular densest-subgraph problem.
- By setting $g(x) = \log x, h(x) = \log\left(\frac{x(x-1)}{2}\right), \alpha = 1$, the optimal (g, h, α)-edge-surplus problem becomes equivalent to maximizing the edge density $\delta(S) = e[S]/\binom{|S|}{2}$.

No general statements on the complexity characterization of the optimal (g, h, α)-edge-surplus problem can be made, since certain cases are polynomial-time solvable whereas others are **NP**-hard. However, the following theorem provides a family of optimal (g, h, α)-edge-surplus problems that are efficiently solvable.

THEOREM 1. *If $g(x) = x$ and $h(x)$ is a concave function, then the optimal (g, h, α)-edge-surplus problem is in* **P**.

PROOF. The optimal (g, h, α)-edge-surplus problem becomes $\max_{\emptyset \neq S \subseteq V} e[S] - \alpha h(|S|)$ where $h(x)$ is a concave function. The claim follows directly from the following succession of facts.
Fact 1: The function defined by the map $S \mapsto e[S]$ is a supermodular function.
Fact 2: The function $h(|S|)$ is submodular given that h is concave. Since $\alpha > 0$, the function $-\alpha h(|S|)$ is supermodular.
Fact 3: Combining the above facts with the fact that the sum of two supermodular functions is supermodular, we obtain that $f_\alpha(S)$ is a supermodular function.
Fact 4: Maximizing supermodular functions is strongly polynomial-time solvable [28]. □

Finally, an important property of the edge-surplus abstraction is that it allows us to model scenarios in numerous practical situations where one wants to find a dense subgraph with bounds on its size. For instance, by relaxing the monotonicity property of h, the k-densest subgraph problem can be modeled as an optimal (g, h, α)-edge-surplus problem by setting $g(x) = x$ and

$$h(x) = \begin{cases} 0, & x = k \\ +\infty, & \text{otherwise.} \end{cases}$$

By choosing $h(x)$ appropriately, one can design algorithms that avoid outputting subgraphs of undesired size.

3. OPTIMAL QUASI-CLIQUES

By setting $g(x) = x$, $h(x) = \frac{x(x-1)}{2}$, and restricting $\alpha \in (0, 1)$ in Problem 1, we obtain the problem we address in this paper, which we call OQC-PROBLEM.

PROBLEM 2 (OQC-PROBLEM). *Given a graph $G = (V, E)$, find a subset of vertices $S^* \subseteq V$ such that*

$$f_\alpha(S^*) = e[S] - \alpha\binom{|S|}{2} \geq f_\alpha(S), \textit{for all } S \subseteq V.$$

We refer to the set S^* as the optimal quasi-clique of G.

3.1 Problem characterization

Hardness. Theorem 1 shows a class of problems which are solvable in polynomial time, while leaving open the hardness characterization of the problems that do not fall into that class. Our OQC-PROBLEM belongs to the latter class of problems: it is not among the polynomial-time solvable optimal (g, h, α)-edge-surplus problems stated in Theorem 1, thus any result about its hardness is not immediate. However, in this regard, we note the following.

For any single edge (u, v), $f_\alpha(\{u, v\}) > 0$; but, for any set S, where $|S|$ is large enough and $e[S] = \alpha\binom{|S|}{2}$, $f_\alpha(S) = 0$. Therefore, the OQC-PROBLEM assigns a positive score to sets S which have density strictly greater than α. Specifically, let $\mathcal{F} = \{S_1, \ldots, S_k\}$ be the family of sets such that $f_\alpha(S_i) > 0$, for all $S_i \in \mathcal{F}$. Notice that if the input graph G is connected, then $k \geq 1$, as $f_\alpha(\{u, v\}) > 0$, for any edge (u, v). This suggests that $e[S_i] = (\alpha + \epsilon_i)\binom{|S_i|}{2}, \epsilon_i > 0$ for all $i = 1, \ldots, k$. The objective of the OQC-PROBLEM is equivalent to maximizing over all sets in $\mathcal{F}$ the product $\epsilon_i\binom{|S_i|}{2}$. In conclusion, therefore, the OQC-PROBLEM is closely related to the problem of finding a maximum clique in a graph, thus being suspected to be **NP**-hard. However, a formal proof of hardness is nontrivial and it constitutes an interesting open problem for future research.

Parameter selection. A natural question that arises whenever a parameter exists is how to choose an appropriate value. We provide here a simple empirical criterion to properly pick the α parameter in our f_α function.

Algorithm 1 GreedyOQC

Input: Graph $G(V, E)$
Output: Subset of vertices $\bar{S} \subseteq V$
  $S_n \leftarrow V$
  for $i \leftarrow n$ downto 1 **do**
    Let v be the vertex with the smallest degree in $G[S_i]$
    $S_{i-1} \leftarrow S_i \setminus \{v\}$
  end for
  $\bar{S} \leftarrow \arg\max_{i=1,\ldots,n} f_\alpha(S_i)$

Let us consider two disjoint sets of vertices S_1, S_2 in the graph G. Assume that $G[S_1 \cup S_2]$ is disconnected, i.e., $G[S_1]$ and $G[S_2]$ form two separate connected components. Also, without any loss of generality, assume that $f_\alpha(S_1) \leq f_\alpha(S_2)$. As our goal is to favor small dense subgraphs, a natural condition to satisfy is $f_\alpha(S_1 \cup S_2) \leq f_\alpha(S_1) \leq f_\alpha(S_2)$, i.e., we require for our objective to prefer the set S_1 (or S_2) rather than the larger set $S_1 \cup S_2$. Therefore, we obtain:

$$e[S_1] + e[S_2] - \alpha\binom{|S_1|+|S_2|}{2} \leq e[S_1] - \alpha\binom{|S_1|}{2},$$

which, considering that $e[S_2] \leq \binom{|S_2|}{2}$, leads to:

$$\alpha \geq \frac{\binom{|S_2|}{2}}{\binom{|S_1|+|S_2|}{2} - \binom{|S_1|}{2}} = \frac{|S_2| - 1}{2|S_1| + |S_2| - 1}.$$

Let us now assume for simplicity that $|S_1| = |S_2| = k$; then the above condition becomes: $\alpha \geq \frac{k-1}{3k-1}$. As $\frac{k-1}{3k-1} < \frac{1}{3}$, it suffices choosing $\alpha \geq \frac{1}{3}$ to have the condition satisfied.

Thus, we choose a value for α around $\frac{1}{3}$, which is actually the value we adopt in our experiments. Alternatively, one could choose $\alpha = e[V]/\binom{|V|}{2}$ to obtain a normalized version of our objective. However, we do not advocate this choice since typically $e[V] = o(|V|^2)$.

3.2 Algorithms

A greedy approximation algorithm. The first efficient algorithm we propose is an adaptation of the greedy algorithm by Asashiro et al. [6], which has been shown to provide a $\frac{1}{2}$-approximation for the densest subgraph problem [10]. The outline of our algorithm, called GreedyOQC, is shown as Algorithm 1. The algorithm iteratively removes the vertex with the smallest degree. The output is the subgraph produced over all iterations that maximizes the objective function f_α. The algorithm can be implemented in $\mathcal{O}(n+m)$ time: the trick consists in keeping a list of vertices for each possible degree and updating the degree of any vertex v during the various iterations of the algorithm simply by moving v to the appropriate degree list.

The GreedyOQC algorithm provides an additive approximation guarantee for the OQC-Problem, as shown next.

Theorem 2. *Let $\bar{S}$ be the set of vertices outputted by the* GreedyOQC *algorithm and let S^* be the optimal vertex set. Consider also the specific iteration of the algorithm where a vertex within S^* is removed for the first time and let S_I denote the vertex set currently kept in that iteration. It holds that:*

$$f_\alpha(\bar{S}) \geq f_\alpha(S^*) - \frac{\alpha}{2}|S_I|(|S_I| - |S^*|).$$

Proof. Given a subset of vertices $S \subseteq V$ and a vertex $u \in S$, let $d_S(u)$ denote the degree of u in $G[S]$.

We start the analysis by considering the first vertex belonging to S^* removed by the algorithm from the current vertex set. Let v denote such a vertex, and let also S_I denote the set of vertices still present just before the removal of v. By the optimality of S^*, we obtain:

$$\begin{aligned}
&f_\alpha(S^*) \geq f_\alpha(S^* \setminus \{u\}),\ \forall u \in S^* \\
&\Leftrightarrow\ e[S^*] - \alpha\binom{|S^*|}{2} \geq (e[S] - d_{S^*}(u)) - \alpha\binom{|S^*|-1}{2}, \forall u \in S^* \\
&\Leftrightarrow\ d_{S^*}(u) \geq \alpha(|S^*| - 1),\ \forall u \in S^*.
\end{aligned}$$

As the algorithm greedily removes vertices with the smallest degree in each iteration, it is easy to see that $d_V(u) \geq d_{S_I}(u) \geq d_{S^*}(u) \geq \alpha(|S^*| - 1)$, $\forall u$. Therefore, noticing also that $S^* \subseteq S_I$, it holds that:

$$\begin{aligned}
f_\alpha(S_I) &= e[S_I] - \alpha\binom{|S_I|}{2} \\
&= \frac{1}{2}\left(\sum_{u \in S^*} d_{S^*}(u) + \sum_{u \in S^*}(d_{S_I}(u) - d_{S^*}(u)) + \right. \\
&\quad \left. + \sum_{u \in S_I \setminus S^*} d_{S_I}(u)\right) - \alpha\binom{|S_I|}{2} \\
&\geq \frac{1}{2}\left(\sum_{u \in S^*} d_{S^*}(u) + \sum_{u \in S_I \setminus S^*} d_{S_I}(u)\right) - \alpha\binom{|S_I|}{2} \\
&= e[S^*] + \frac{1}{2}\sum_{u \in S_I \setminus S^*} d_{S_I}(u) - \alpha\binom{|S_I|}{2} \\
&\geq e[S^*] + \frac{1}{2}(|S_I| - |S^*|)\alpha(|S^*| - 1) - \alpha\binom{|S_I|}{2} \\
&= f_\alpha(S^*) - \frac{\alpha}{2}|S_I|(|S_I| - |S^*|).
\end{aligned}$$

As the final output of the algorithm is the best over all iterations, we finally obtain:

$$f_\alpha(\bar{S}) \ \geq\ f_\alpha(S_I) \ \geq\ f_\alpha(S^*) - \frac{\alpha}{2}|S_I|(|S_I| - |S^*|).$$

□

The above result can be interpreted as follows. Assuming that $|S_I|$ is $\mathcal{O}(|\bar{S}|)$, the additive approximation factor proved in Theorem 2 becomes $f_\alpha(\bar{S}) \geq f_\alpha(S^*) - \frac{\alpha}{2}|\bar{S}|(|\bar{S}| - |S^*|)$. Thus, the error achieved by the GreedyOQC algorithm is guaranteed to be bounded by an additive factor proportional to the size of the optimal quasi-clique outputted. As optimal quasi-cliques are typically small graphs, this results in an approximation guarantee that is very tight in practice.

A local-search heuristic. Even though the above GreedyOQC algorithm achieves provable approximation guarantee, it is not guaranteed for that algorithm to be related to any (local) optimal solution, which is a desirable property that in many practical cases can lead to very good results. To this purpose, we present next a local-search heuristic, called LocalSearchOQC, which performs local operations and outputs a vertex set S that is guaranteed to be locally optimal, i.e., if any single vertex is added to or removed from S, then the objective function decreases.

Algorithm 2 LocalSearchOQC

Input: Graph $G = (V, E)$; maximum number of iterations T_{MAX}
Output: Subset of vertices $\bar{S} \subseteq V$

```
S ← {v}, where v is chosen uniformly at random
b1 ← TRUE, t ← 1.
while b1 and t ≤ T_MAX do
  b2 ← TRUE
  while b2 do
    If there exists u ∈ V\S such that f_α(S∪{u}) ≥ f_α(S)
    then let S ← S ∪ {u}
    otherwise set b2 ← FALSE
  end while
  If there exists u ∈ S such that f_α(S\{u}) ≥ f_α(S)
  then let S ← S\{u}
  otherwise, set b1 ← FALSE
  t ← t + 1
end while
S̄ ← arg max_{Ŝ∈{S,V\S}} f_α(Ŝ)
```

The outline of LocalSearchOQC is shown as Algorithm 2. The algorithm initially selects a random vertex and then it keeps adding vertices to the current set S while the objective improves. When no vertices can be added, the algorithm tries to find a vertex in S whose removal may improve the objective. As soon as such a vertex is encountered, it is removed from S and the algorithm re-starts from the adding phase. The process continues until a local optimum is reached or the number of iterations exceeds $T_{\max}$. The time complexity of LocalSearchOQC is $\mathcal{O}(T_{\max}\, m)$.

The effectiveness of the LocalSearchOQC algorithm partly depends on the initial seeding set S. To this end, we devise a heuristic to choose an initial seeding set more appropriately than setting it equal to a randomly selected vertex. Let v^* be the vertex that maximizes the ratio $\frac{t(v^*)}{d(v^*)}$, where $t(v^*)$ is the number of triangles of v^* and $d(v^*)$ its degree (we approximate the number of triangles in which each vertex participates with the technique described in [24]). Given vertex v^*, we use as a seed the set $\{v^* \cup N(v^*)\}$, where $N(v^*) = \{u : (u, v^*) \in E\}$ is the neighborhood of v^*.

4. PROBLEM VARIANTS

We present here two variants of our basic problem, that have many practical applications: finding top-k optimal quasi-cliques (Section 4.1) and finding an optimal quasi-clique that contains a given set of *query vertices* (Section 4.2).

4.1 Top-k optimal quasi-cliques

The top-k version of our problem is as follows: given a graph $G = (V, E)$ and a constant k, find top-k *disjoint* optimal quasi-cliques. This variant is particularly useful in scenarios where finding a single dense subgraph is not sufficient, rather a set of $k > 1$ dense components is required.

From a formal viewpoint, the problem would require to find k subgraphs for which the sum of the various objective function values computed on each subgraph is maximized. Due to its intrinsic hardness, however, here we heuristically tackle the problem in a greedy fashion: we find one dense subgraph at a time, we remove all the vertices of the subgraph from the graph, and we continue until we find k subgraphs or until we are left with an empty graph. Note that this iterative approach allows us to automatically fulfill a very common requirement of finding top-k subgraphs that are pairwise disjoint.

Table 2: Graphs used in our experiments.

	Vertices	Edges	Description
Dolphins	62	159	Biological Network
Polbooks	105	441	Books Network
Adjnoun	112	425	Adj. and Nouns in 'David Copperfield'
Football	115	613	Games Network
Jazz	198	2 742	Musicians Network
Celegans N.	297	2 148	Biological Network
Celegans M.	453	2 025	Biological Network
Email	1 133	5 451	Email Network
AS-22july06	22 963	48 436	Auton. Systems
Web-Google	875 713	3 852 985	Web Graph
Youtube	1 157 822	2 990 442	Social Network
AS-Skitter	1 696 415	11 095 298	Auton. Systems
Wikipedia 2005	1 634 989	18 540 589	Web Graph
Wikipedia 2006/9	2 983 494	35 048 115	Web Graph
Wikipedia 2006/11	3 148 440	37 043 456	Web Graph

4.2 Constrained optimal quasi-cliques

The constrained optimal quasi-cliques variant consists in finding an optimal quasi-clique that contains a set of prespecified *query vertices*. This variant is inspired by the *community-search problem* [30], which has many applications, such as finding thematic groups, organizing social events, tag suggestion. Next, we formalize the problem, prove that it is **NP**-hard, and adapt our algorithms for this variant.

Let $G = (V, E)$ be a graph, and $Q \subseteq V$ be a set of query vertices. We want to find a set of vertices $S \subseteq V$, so that S contains the query vertices Q and maximizes our objective function f_α. Formally, we define the following problem.

Problem 3 (Constrained-OQC-Problem). *Given a graph $G = (V, E)$ and set $Q \subseteq V$, find $S^* \subseteq V$ such that $f_\alpha(S^*) = \max_{Q \subseteq S \subseteq V} f_\alpha(S)$.*

It is easy to see that, when $Q = \emptyset$, the Constrained-OQC-Problem reduces to the OQC-Problem. However, contrarily to the basic OQC-Problem, the Constrained-OQC-Problem can very easily be shown to be **NP**-hard. The hardness is quite immediate from Theorem 1 in [31] and we omit details due to space constraints.

Theorem 3. *The* Constrained-OQC-Problem *is* **NP**-*hard.*

The GreedyOQC algorithm can be adapted to solve the Constrained-OQC-Problem simply by ignoring the nodes $u \in Q$ during the execution of the algorithm, so as to never remove vertices of Q.

Similarly, our LocalSearchOQC algorithm can solve the Constrained-OQC-Problem with a couple of simple modifications: the set S is initialized to the set of query vertices Q, while, during the iterative phase of the algorithm, we never allow a vertex $u \in Q$ to leave S.

5. EXPERIMENTAL EVALUATION

In this section we present our empirical evaluation, first on publicly-available real-world graphs (Section 5.1), whose main characteristics are shown in Table 2, and then on synthetic graphs where the ground truth is known (Section 5.2).

Table 3: Densest subgraphs extracted with Charikar's method vs. optimal quasi-cliques extracted with the proposed GreedyOQC algorithm (greedy) and LocalSearchOQC algorithm (ls). $\delta = e[S]/\binom{|S|}{2}$ is the edge density of the extracted subgraph S, D is the diameter, and $\tau = t[S]/\binom{|S|}{3}$ is the triangle density.

	$\lvert S\rvert$			δ			D			τ		
	densest	opt. quasi-clique		densest	opt. quasi-clique		densest	opt. quasi-clique		densest	opt. quasi-clique	
	subgraph	GREEDY	LS	subgraph	GREEDY	LS	subgraph	GREEDY	LS	subgraph	GREEDY	LS
Dolphins	19	13	8	0.27	0.47	0.68	3	3	2	0.05	0.12	0.32
Polbooks	53	13	16	0.18	0.67	0.61	6	2	2	0.02	0.28	0.24
Adjnoun	45	16	15	0.20	0.48	0.60	3	3	2	0.01	0.10	0.12
Football	115	10	12	0.09	0.89	0.73	4	2	2	0.03	0.67	0.34
Jazz	99	59	30	0.35	0.54	1	3	2	1	0.08	0.23	1
Celeg. N.	126	27	21	0.14	0.55	0.61	3	2	2	0.07	0.20	0.26
Celeg. M.	44	22	17	0.35	0.61	0.67	3	2	2	0.07	0.26	0.33
Email	289	12	8	0.05	1	0.71	4	1	2	0.01	1	0.30
AS-22july06	204	73	12	0.40	0.53	0.58	3	2	2	0.09	0.19	0.20
Web-Google	230	46	20	0.22	1	0.98	3	2	2	0.03	0.99	0.95
Youtube	1874	124	119	0.05	0.46	0.49	4	2	2	0.02	0.12	0.14
AS-Skitter	433	319	96	0.41	0.53	0.49	2	2	2	0.10	0.19	0.13
Wiki '05	24555	451	321	0.26	0.43	0.48	3	3	2	0.02	0.06	0.10
Wiki '06/9	1594	526	376	0.17	0.43	0.49	3	3	2	0.10	0.06	0.11
Wiki '06/11	1638	527	46	0.17	0.43	0.56	3	3	2	0.31	0.06	0.35

Our main goal is to compare our optimal quasi-cliques with densest subgraphs. For extracting optimal quasi-cliques, we involve both our proposed algorithms, i.e., GreedyOQC and LocalSearchOQC, which, following the discussion in Section 3.1, we run with $\alpha = \frac{1}{3}$ (for LocalSearchOQC, we also set $T_{\max} = 50$). For finding densest subgraphs, we use the Goldberg's exact algorithm [19] for small graphs, while for graphs whose size does not allow the Goldberg's algorithm to terminate in reasonable time we use the Charikar's $\frac{1}{2}$-approximation algorithm [10].

All algorithms are implemented in java, and all experiments are performed on a single machine with Intel Xeon cpu at 2.83GHz and 50GB ram.

5.1 Real-world graphs

Results on real graphs are shown in Table 3. We compare optimal quasi-cliques outputted by the proposed GreedyOQC and LocalSearchOQC algorithms with densest subgraphs extracted with the Charikar's algorithm. Particularly, we use the Charikar's method to be able to handle the largest graphs. For consistency, Table 3 reports on results achieved by Charikar's method also for the smallest graphs. We recall that the results in Table 1 in the Introduction refer instead to the exact Goldberg's method. However, a comparison of the two tables on their common rows shows that the Charikar's algorithm, even though it is approximate, produces almost identical results with the results produced by the Goldberg's algorithm.

Table 3 clearly confirms the preliminary results reported in the Introduction: optimal quasi-cliques have larger edge and triangle densities, and smaller diameter than densest subgraphs. Particularly, the edge density of optimal quasi-cliques is evidently larger on all graphs. For instance, on Football and Youtube, the edge density of optimal quasi-cliques (for both the GreedyOQC and LocalSearchOQC algorithms) is about 9 times larger than the edge density of densest subgraphs, while on Email the difference increases up to 20 times (GreedyOQC) and 14 times (LocalSearchOQC). Still, the triangle density of the optimal quasi-cliques outputted by both GreedyOQC and LocalSearchOQC is one order of magnitude larger than the triangle density of densest subgraphs on 11 out of 15 graphs.

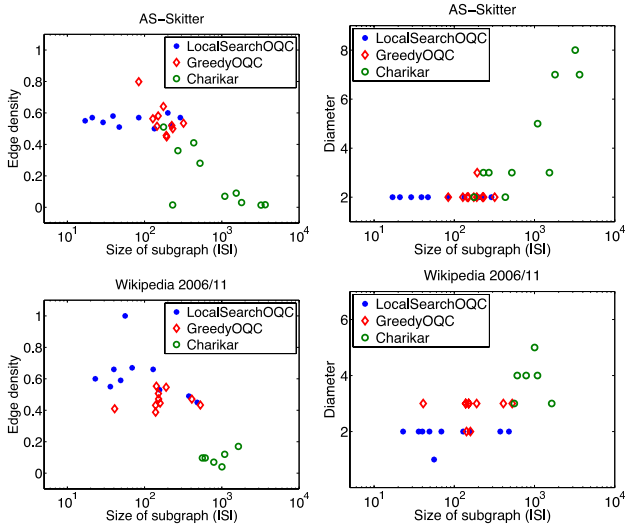

Figure 1: Edge density and diameter of the top-10 subgraphs found by our GreedyOQC and LocalSearchOQC methods, and Charikar's algorithm, on the AS-skitter graph (top) and the Wikipedia 2006/11 graph (bottom).

Comparing our two algorithms to each other, we can see that LocalSearchOQC performs generally better than GreedyOQC. Indeed, the edge density achieved by LocalSearchOQC is higher than that of GreedyOQC on 10 out of 15 graphs, while the diameter of the LocalSearchOQC optimal quasi-cliques is never larger than the diameter of the GreedyOQC optimal quasi-cliques.

Concerning efficiency, all algorithms are linear in the number of edges of the graph. Charikar's and GreedyOQC algorithm are somewhat slower than LocalSearchOQC, but mainly due to bookkeeping. LocalSearchOQC algorithm's running times vary from milliseconds for the small graphs (e.g., 0.004s for Dolphins, 0.002s for Celegans N.), few seconds for the larger graphs (e.g., 7.94s for Web-Google and 3.52s for Youtube) and less than one minute for the largest graphs (e.g., 59.27s for Wikipedia 2006/11).

Top-k optimal quasi-cliques. Figure 1 evaluates top-k optimal quasi-cliques and top-k densest subgraphs on the AS-Skitter and Wikipedia 2006/11 graphs using the iterative method described in Section 4.1. Similar results hold for the other graphs but are omitted due to space constraints.

For each graph we show two scatterplots. The x axis in logarithmic scale reports the size of each of the top-k dense components, while the y axes show the edge density and the diameter, respectively. In all figures, optimal quasi-cliques correspond to blue filled circles (LocalSearchOQC) or red diamonds (GreedyOQC), while densest subgraphs correspond to green circles. It is evident that optimal quasi-cliques are significantly better in terms of both edge density and diameter also in this top-k variant. The edge density is in the range $0.4 - 0.7$ and the diameter is always 2 or 3, except for a 56-vertex clique in Wikipedia 2006/11 with diameter 1. On the contrary, the densest subgraphs are large graphs, with diameter ranging typically from 3 to 5, with significantly smaller edge densities: besides few exceptions, the edge density of densest subgraphs is always around 0.1 or even less.

5.2 Synthetic graphs

Experiments on synthetic graphs deal with the following task: a (small) clique is planted in two different types of random graphs, and the goal is to check if the dense subgraph algorithms are able to recover those cliques. Two different random-graph models are used as host graphs for the cliques: (i) Erdős-Rényi and (ii) random power-law graphs. In the former model, each edge exists with probability p independently of the other edges. To generate a random power-law graph, we follow the Chung-Lu model [11]: we first generate a degree sequence $(d_1, \ldots, d_n)$ that follows a power law with a pre-specified slope and we connect each pair of vertices i, j with probability proportional to $d_i d_j$.

We evaluate our algorithms by measuring how "close" are the returned subgraphs to the planted clique. In particular, we use the measures of *precision* P and *recall* R, defined as

$$P = \frac{\#\{\text{returned vertices from hidden clique}\}}{size\{\text{subgraph returned}\}}, \text{ and}$$

$$R = \frac{\#\{\text{returned vertices from hidden clique}\}}{size\{\text{hidden clique}\}}.$$

Next we discuss the results obtained. For the Erdős-Rényi model we also provide a theoretical justification of the outcome of the two tested algorithms.

Erdős-Rényi graphs. We plant a clique of 30 vertices on Erdős-Rényi graphs with $n = 3\,000$ and edge probabilities $p \in \{0.5, 0.1, 0.008\}$. Those values of p are selected to represent very dense, medium-dense, and sparse graphs.

We report in Table 4 the results of running our LocalSearchOQC and GreedyOQC algorithms for extracting optimal quasi-cliques, as well as the Goldberg's algorithm for extracting densest subgraphs. We observe that our two algorithms, LocalSearchOQC and GreedyOQC, produce *identical* results, thus we refer to both of them as optimal quasi-cliques algorithms. We see that the algorithms produce two kinds of results: they either find the hidden clique, or they miss it and return the whole graph. In the very dense setting ($p = 0.5$) all algorithms miss the clique, while in the sparse setting ($p = 0.008$) all algorithms recover it. However, at the middle-density setting ($p = 0.1$) only the optimal quasi-cliques algorithms find the clique, while the Goldberg's algorithm misses it.

Table 4: Subgraphs returned by the Goldberg's max-flow algorithm and by our two algorithms (GreedyOQC, LocalSearchOQC) on Erdős-Rényi graphs with 3 000 vertices and three values of p, and with a planted clique of 30 vertices.

Erdős-Rényi parameters		densest subgraph			optimal quasi-clique		
n	p	$\lvert S\rvert$	P	R	$\lvert S\rvert$	P	R
3 000	0.5	3 000	0.01	1.00	3 000	0.01	1.00
3 000	0.1	3 000	0.01	1.00	30	1.00	1.00
3 000	0.008	30	1.00	1.00	30	1.00	1.00

To better understand the results shown on Table 4, we provide a theoretical explanation of the behavior of the algorithms depending on their objective. Assume that h is the size of the hidden clique, $p > \log n/n$. If $np \geq h - 1$ the densest subgraph criterion always returns the whole graph with high probability. In our experiments, this happens with $p = 0.5$ and $p = 0.1$. On the other hand, if $np < h - 1$, the densest subgraph corresponds to the hidden clique, and therefore the Goldberg's algorithm cannot miss it.

Now consider our objective function , i.e., the edge-surplus function f_α. The expected score for the hidden clique is $\mathbb{E}[f_\alpha(H)] = f_\alpha(H) = (1-\alpha)\binom{h}{2}$. The expected score for the whole network is $\mathbb{E}[f_\alpha(V)] = \left(p\binom{n}{2} + (1-p)\binom{h}{2}\right) - \alpha\binom{n}{2}$. We obtain the following two cases: (A) when $p > \alpha$, we have $\mathbb{E}[f_\alpha(V)] \geq f_\alpha(H)$. (B) when $p < \alpha$, we have $f_\alpha(H) \geq \mathbb{E}[f_\alpha(V)]$. This rough analysis explains our findings.

Power-law graphs. We plant a clique of 15 vertices in random power-law graphs of again 3 000 vertices, with power-law exponent varying from 2.2 to 3.1. We select these values since most real-world networks have power-law exponent in this range [27]. For each exponent tested, we generate five random graphs, and all the figures we report are averages over these five trials.

Again, we compare our GreedyOQC and LocalSearchOQC algorithms with the Goldberg's algorithm. The LocalSearchOQC algorithm is run seeded with one of the vertices of the clique. The justification of this choice is that one can always re-run the algorithm until it finds such a vertex with high probability.[2]

The precision and recall scores of the three competing algorithms as a function of the power-law exponent are shown in Figure 2. As the exponent increases the host graph becomes sparser and both algorithms have no difficulties in finding the hidden clique. However, for exponent values ranging between 2.2 and 2.6 the optimal quasi-cliques are significantly better than the densest subgraphs. Indeed, in terms of precision, the Goldberg's algorithm is outperformed by both our algorithms. In terms of recall, our LocalSearchOQC is better than Goldberg's, while our GreedyOQC performs slightly worse. An explanation for this is that the GreedyOQC algorithm detects other high-density subgraphs, but not exactly the planted clique. As an ex-

[2] If the hidden clique is of size $\mathcal{O}(n^\epsilon)$, for some $0 \leq \epsilon < 1$, it suffices to run the algorithm a sub-linear number of times (i.e., $\mathcal{O}((1-\gamma)n^{1-\epsilon})$ times) in order to obtain one of the vertices of the clique as a seed with probability at least $1 - \gamma$.

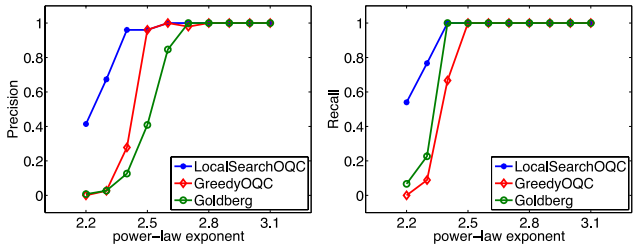

Figure 2: Precision and recall for our method and Goldberg's algorithm vs. the power-law exponent of the host graph.

ample, with power-law exponent 2.3, GREEDYOQC finds a subgraph with 23 vertices and edge density 0.87.

Stability with respect to α. We also test the sensitivity of our density measure with respect to the parameter α. We use again the planted-clique setting, and we test the ability of our algorithms to recover the clique as we vary the parameter α. We omit detailed plots, due to space constraints, but we report that the behavior of both algorithms is extremely stable with respect to α. Essentially, the algorithms again either find the clique or miss it, depending on the graph-generation parameters, as we saw in the previous section, namely, the probability p of the Erdős-Rényi graphs, or the exponent of the power-law graphs. Moreover, in all cases, the performance of our algorithms, measured by precision and recall as in the last experiment, does not depend on α.

6. APPLICATIONS

In this section we show experiments concerning our constrained optimal quasi-cliques variant introduced in Section 4.2. To this end, we focus on two applications that can be commonly encountered in real-world scenarios: finding thematic groups and finding highly-correlated genes from a microarray dataset. For the sake of brevity of presentation, we show next results for only one of our algorithms, particularly the LOCALSEARCHOQC algorithm.

6.1 Thematic groups

Motivation. Suppose that a set of scientists Q wants to organize a workshop. How do they invite other scientists to participate in the workshop so that the set of all the participants, including Q, have similar interests?

Setup. We use a co-authorship graph extracted from the DBLP dataset. The dataset contains publications in all major computer-science journals. There is an undirected edge between two authors if they have coauthored a journal article. Taking the largest connected component gives a graph of 226K vertices and 1.4M edges.

We evaluate the results of our algorithm qualitatively, in a sanity check form rather than a strict and quantitative way, which is not even well-defined. We perform the following two queries: $Q_1 = \{\text{Papadimitriou}, \text{Abiteboul}\}$ and $Q_2 = \{\text{Papadimitriou}, \text{Blum}\}$.

Results. Papadimitriou is one of the most prolific computer scientists and has worked on a wide range of areas. With query Q_1 we invoke his interests in database theory given that Abiteboul is an expert in this field. As we can observe from Figure 3, the optimal quasi-clique outputted contains

Abiteboul, Bernstein, Brodie, Carey, Ceri, Crof, DeWitt, Ehrenfeucht, Franklin, Gawlick, Gray, Haas, Halevy, Hellerstein, Ioannidis, Jagadish, Kanellakis, Kersten, Lesk, Maier, Molina, Naughton, Papadimitriou, Pazzani, Pirahesh, Schek, Sellis, Silberschatz, Snodgrass, Stonebraker, Ullman, Weikum, Widom, Zdonik

Figure 3: Authors returned by our LOCALSEARCHOQC algorithm when queried with Papadimitriou and Abiteboul. The set includes well-known database scientists. The induced subgraph has 34 vertices and 457 edges. The edge density is 0.81, the diameter is 3, the triangle density is 0.66.

Alt, Blum, Garey, Guibas, Johnson, Karp, Mehlhorn, Papadimitriou, Preparata, Tarjan, Welzl, Widgerson, Yannakakis,

Figure 4: Authors returned by our LOCALSEARCHOQC algorithm when queried with Papadimitriou and Blum. The set includes well-known theoretical computer scientists. The induced subgraph has 13 vertices and 38 edges. The edge density is 0.49, the diameter is 3, the triangle density is 0.14.

database scientists. On the other hand, with query Q_2 we invoke Papadimitriou's interests in theory, given that Blum is a Turing-award theoretical computer scientist. As we can see in Figure 4, the returned optimal quasi-clique contains well-known theoretical computer scientists.

6.2 Correlated genes

Motivation. Detecting correlated genes has several applications. For instance, clusters of genes with similar expression levels are typically under similar transcriptional control. Furthermore, genes with similar expression patterns may imply co-regulation or relationship in functional pathways. Detecting gene correlations has played a key role in discovering unknown types of breast cancer [29]. Here, we wish to illustrate that optimal quasi-cliques provide a useful graph-theoretic framework for gene co-expression network analysis [25], without delving deeply into biological aspects of the results.

Setup. We use the publicly-available breast-cancer dataset of van de Vijner et al. [32], which consists of measurements across 295 patients of 24 479 probes. Upon running a standard probe-selection algorithm based on Singular Value Decomposition (SVD), we obtain a 295×1000 matrix. The graph G in input to our LOCALSEARCHOQC algorithm is derived using the well-established approach defined in [25]: each gene corresponds to a vertex in G, while an edge between any pair of genes i, j is drawn if and only if the modulus of the Pearson's correlation coefficient $|\rho(i, j)|$ exceeds a given threshold θ ($\theta = 0.99$ in our setting). We perform the following query, along the lines of the previous section: "find highly-correlated genes with the tumor protein 53 (p53)". We select p53 as it is known to be central in tumorigenesis.

Results. The output of our algorithm is a clique consisting of 14 genes shown in Figure 5. A potential explanation of our finding is the pathway depicted in Figure 6, which shows that the activation of the p53 signaling can be initiated by signals coming from the PI3K/AKT pathway. Both PI3KCA and

p53, BRCA1, ARID1A, ARID1B, ZNF217, FGFR1, KRAS, NCOR1, PIK3CA, APC, MAP3K13, STK11, AKT1, RB1

Figure 5: Genes returned by our LocalSearchOQC algorithm when queried with p53. The induced subgraph is a clique with 14 vertices.

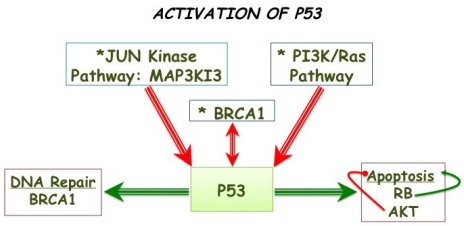

Figure 6: A tumorigenesis pathway consistent with our findings.

AKT1 that are detected by our method are key players of this pathway. Furthermore, signals from the JUN kinase pathway can also trigger the p53-cascade; MAP3K13 is a member of this pathway.

One of the results of p53 signaling is apoptosis, a process promoted by RB. The latter can also regulate the stability and the apoptotic function of p53. Finally, our output includes BRCA1, which is known to physically associate with p53 and affect its actions [33].

7. CONCLUSIONS

In this work we introduce a novel density measure to extract high-quality subgraphs. We show that the proposed density function is included into a more general framework for dense-subgraph extraction, which also subsumes other various popular density functions and provides a principled way to derive application-specific algorithms and heuristics. We provide theoretical insights both into the general framework and in the proposed function. We design two efficient algorithms to optimize our function: an additive approximation algorithm, as well as a local-search heuristic. We test our algorithms on real graphs, showing that the subgraphs outputted by our methods have larger edge and triangle densities, and smaller diameter than the subgraphs extracted by the method that optimizes the popular average-degree measure. We also evaluate our methods in tackling a couple of variants of our original problem, i.e., finding top-k dense subgraphs and finding subgraphs containing a set of pre-specified vertices, as well as on real-world data-mining and bioinformatics applications, such as forming thematic groups and finding highly-correlated genes from a microarray dataset.

Our work leaves several open problems, such as the formal hardness characterization of our OQC-Problem, the formal analysis of LocalSearchOQC, the design of efficient randomized algorithms, and the derivation of more densities from the general framework with desirable properties for existing applications.

8. REFERENCES

[1] J. Abello, M. G. C. Resende, and S. Sudarsky. Massive quasi-clique detection. In *LATIN*, 2002.

[2] R. Andersen and K. Chellapilla. Finding dense subgraphs with size bounds. In *WAW*, 2009.

[3] A. Angel, N. Sarkas, N. Koudas, and D. Srivastava. Dense subgraph maintenance under streaming edge weight updates for real-time story identification. *PVLDB*, 5(6), 2012.

[4] S. Arora, D. Karger, and M. Karpinski. Polynomial time approximation schemes for dense instances of NP-hard problems. In *STOC*, 1995.

[5] Y. Asahiro, R. Hassin, and K. Iwama. Complexity of finding dense subgraphs. *Discr. Ap. Math.*, 121(1-3), 2002.

[6] Y. Asahiro, K. Iwama, H. Tamaki, and T. Tokuyama. Greedily finding a dense subgraph. *J. Algorithms*, 34(2), 2000.

[7] C. Bron and J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *CACM*, 16(9), 1973.

[8] M. Brunato, H. H. Hoos, and R. Battiti. On effectively finding maximal quasi-cliques in graphs. In *Learning and Intelligent Optimization*. 2008.

[9] G. Buehrer and K. Chellapilla. A scalable pattern mining approach to web graph compression with communities. In *WSDM*, 2008.

[10] M. Charikar. Greedy approximation algorithms for finding dense components in a graph. In *APPROX*, 2000.

[11] F. R. K. Chung and L. Lu. The average distance in a random graph with given expected degrees. *Internet Mathematics*, 1(1), 2003.

[12] X. Du, et al. Migration motif: a spatial - temporal pattern mining approach for financial markets. In *KDD*, 2009.

[13] U. Feige. Approximating maximum clique by removing subgraphs. *SIAM Journal of Discrete Mathematics*, 18(2), 2005.

[14] U. Feige, G. Kortsarz, and D. Peleg. The dense k-subgraph problem. *Algorithmica*, 29(3), 2001.

[15] U. Feige and M. Langberg. Approximation algorithms for maximization problems arising in graph partitioning. *J. Algorithms*, 41(2), 2001.

[16] E. Fratkin, B. T. Naughton, D. L. Brutlag, and S. Batzoglou. MotifCut: regulatory motifs finding with maximum density subgraphs. In *ISMB*, 2006.

[17] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan. A fast parametric maximum flow algorithm and applications. *Journal of Computing*, 18(1), 1989.

[18] D. Gibson, R. Kumar, and A. Tomkins. Discovering large dense subgraphs in massive graphs. In *VLDB*, 2005.

[19] A. V. Goldberg. Finding a maximum density subgraph. Technical report, University of California at Berkeley, 1984.

[20] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182(1), 1999.

[21] R. Jin, Y. Xiang, N. Ruan, and D. Fuhry. 3-hop: a high-compression indexing scheme for reachability query. In *SIGMOD*, 2009.

[22] S. Khot. Ruling out PTAS for graph min-bisection, dense k-subgraph, and bipartite clique. *Journal of Computing*, 36(4), 2006.

[23] S. Khuller and B. Saha. On Finding Dense Subgraphs. *ICALP*, 2009.

[24] M. N. Kolountzakis and et al.: Efficient triangle counting in large graphs via degree-based vertex partitioning. *Internet Mathematics*, 8(1-2), 2012.

[25] M. A. Langston and et al. A combinatorial approach to the analysis of differential gene expression data: The use of graph algorithms for disease prediction and screening. *Methods of Microarray Data Analysis IV*. 2005.

[26] V. E. Lee, N. Ruan, R. Jin, and C. C. Aggarwal. A survey of algorithms for dense subgraph discovery. *Managing and Mining Graph Data*. 2010.

[27] M. Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

[28] A. Schrijver. *Combinatorial Optimization : Polyhedra and Efficiency (Algorithms and Combinatorics)*. Springer, 2004.

[29] T. Sorlie and et al. Repeated observation of breast tumor subtypes in independent gene expression data sets. *PNAS*, 100(14), 2003.

[30] M. Sozio and A. Gionis. The community-search problem and how to plan a successful cocktail party. *KDD*, 2010.

[31] T. Uno. An efficient algorithm for solving pseudo clique enumeration problem. *Algorithmica*, 56(1), 2010.

[32] M. J. van de Vijver and et al. A gene-expression signature as a predictor of survival in breast cancer. *The New England journal of medicine*, 347(25), 2002.

[33] R. A. Weinberg. *The Biology of Cancer HB*. Garland Science, 2006.