Robust Local Community Detection: On Free Rider Effect and Its Elimination

Yubao Wu [1], Ruoming Jin [2], Jing Li [1], Xiang Zhang [1]
[1]Department of Electrical Engineering and Computer Science, Case Western Reserve University
[2]Department of Computer Science, Kent State University
yubao.wu@case.edu; jin@cs.kent.edu; {jingli, xiang.zhang}@case.edu

ABSTRACT

Given a large network, local community detection aims at finding the community that contains a set of query nodes and also maximizes (minimizes) a goodness metric. This problem has recently drawn intense research interest. Various goodness metrics have been proposed. However, most existing metrics tend to include irrelevant subgraphs in the detected local community. We refer to such irrelevant subgraphs as free riders. We systematically study the existing goodness metrics and provide theoretical explanations on why they may cause the free rider effect. We further develop a query biased node weighting scheme to reduce the free rider effect. In particular, each node is weighted by its proximity to the query node. We define a query biased density metric to integrate the edge and node weights. The query biased densest subgraph, which has the largest query biased density, will shift to the neighborhood of the query nodes after node weighting. We then formulate the query biased densest connected subgraph (QDC) problem, study its complexity, and provide efficient algorithms to solve it. We perform extensive experiments on a variety of real and synthetic networks to evaluate the effectiveness and efficiency of the proposed methods.

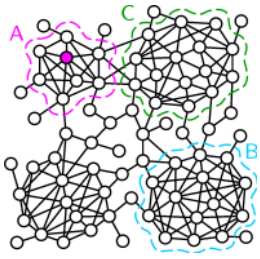
1. INTRODUCTION

Graph (or network[1]) is a ubiquitous data structure that can model many real-world problems. Community detection is a fundamental problem in network analysis [11]. Traditional community detection methods aim at finding all communities in the entire network.

With the increasing size of the networks, local community detection has recently drawn intensive research interest [20, 2, 7, 29, 31]. Given a set of query nodes, the local community detection problem aims at finding the community near the query nodes. It has a wide range of applications in analyzing social networks, collaborative tagging systems, query-logs, and biological networks [31, 9, 20, 19].

[1]In this paper, we use network and graph interchangeably.

Proceedings of the VLDB Endowment, Vol. 8, No. 7



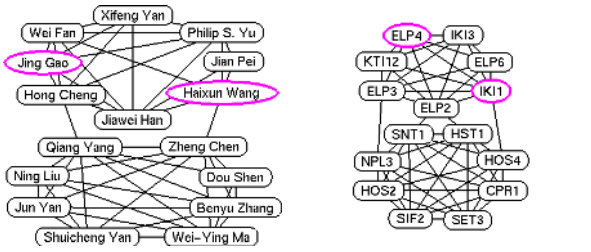
Goodness metrics	A	A∪B	A∪C
Classic density	2.50	2.95	2.83
Edge-surplus	15.3	26.5	22.8
Minimum degree	4	4	4
Subg. modularity	2.0	3.6	4.6
Density-isolation	–2.6	3.8	1.5
Ext. conductance	0.25	0.14	0.11
Local modularity	0.63	0.70	0.78

Figure 1: Left: a network with 4 communities, the query node is the purple node in community A; right: goodness values of different metrics on subgraphs A, A∪B, and A∪C

In local community detection, a goodness metric is usually used to measure whether a subgraph forms a community. The existing goodness metrics for local community detection can be categorized into three classes. The first class optimizes the internal denseness of a subgraph, i.e., the set of nodes in a community should be densely connected with each other. Such metrics include the classic density definition [29], edge-surplus [36], and minimum degree [31]. The second class optimizes both the internal denseness and the external sparseness. That is, the set of nodes in the community are not only densely connected with each other, but also sparsely connected with the nodes that are not in the community. Such metrics include subgraph modularity [23], density-isolation [22], and external conductance [2]. The local modularity measures the sharpness of the community boundary and belongs to the third class [7]. Using this metric, the set of nodes in the boundary of the community are highly connected to the nodes in the community but sparsely connected to the nodes outside the community.

Given a goodness metric, the generic local community detection problem aims at finding a subgraph such that: (1) the subgraph contains the query nodes, and (2) the goodness metric is maximized (or minimized). This formulation has been explicitly adopted in many existing works [31, 9, 24]. Other works can also fit into this form [29, 36, 7, 23, 2]. The local community detection problem has also been studied as the local clustering problem or the community search problem in the literature [33, 31].

Most of the existing goodness metrics tend to include irrelevant subgraphs in the identified local communities. For example, Figure 1 shows a network containing four communities. Suppose that the query node is the purple node in community A. Intuitively, subgraph A should be the target local community. Suppose that we use the classic density definition, which measures the average degree of the nodes in the subgraph. If we merge subgraphs B or C into A, the



(a) Co-author networks (b) Biological networks

Figure 2: Two local communities with free riders identified from the real-world networks using the classic density definition. The query nodes are in purple ellipses.

density will increase: the densities of subgraphs A, A∪B, and A∪C are 2.50, 2.95, and 2.83 respectively. We refer to the irrelevant subgraphs, such as B and C, as free riders in local community detection.

Other goodness metrics also cause the free rider effect. The table on the right in Figure 1 shows the goodness values for subgraphs A, A∪B, and A∪C for different metrics. For all these metrics, merging B or C into A will result in better goodness values. Note that the external conductance tries to minimize the goodness value instead of maximizing it.

In the example shown in Figure 1, subgraph B is the densest subgraph in the entire network. We refer to such a subgraph with the largest goodness value as the global optimal subgraph. If a goodness metric will include the global optimal subgraph in the identified local community, we say this metric causes the global free rider effect. We refer to a subgraph whose goodness value is greater than that of any subgraph of it as a local optimal subgraph. In this figure, subgraph C is a local dense subgraph, which is irrelevant to the query node. If a goodness metric will include a local optimal subgraph in the identified local community, we say that this metric causes the local free rider effect.

In addition to the running example in Figure 1, Figure 2 shows two examples of free riders from real-world networks. Figure 2(a) shows the local community identified in the co-author network extracted from the DBLP dataset. The query nodes are in purple ellipses. The classic density is used as the goodness metric. It is clear from this figure that the lower part is a free rider that should not have been included in the result. Figure 2(b) shows the local community identified in the protein interaction network downloaded from BioGRID (*thebiogrid.org*). The query proteins {ELP4, IKI1} are from the protein complex transcription elongation factor [13]. The lower part in the figure does not belong to the complex and clearly is a free rider. Similar results can be observed when using other metrics listed in the table in Figure 1.

In this paper, we systematically study the existing goodness metrics and provide theoretical explanations on why they will cause global or local free rider effects. To reduce the free rider effect, we propose a node weighting scheme based on random walk. The nodes far away from the query nodes will have large weights and high penalties to be included in the target local community. This query biased weighting scheme forces the global densest subgraph shift to the neighborhood of the query nodes. We use the query biased density as the new metric and formulate the query biased densest connected subgraph (QDC) problem. Its goal is to find the query biased densest subgraph that contains the query nodes and is also connected.

The original QDC problem is NP-hard. To solve it efficiently, we study two related problems QDC′ and QDC″. In QDC″, we only maximize the query biased density. In QDC′, in addition to maximizing the query biased density, we also have the constraint that the query nodes must be included in the resulting subgraph. These two problems can be solved in polynomial time. Moreover, the solutions to these two problems can often be used as the optimal or approximate solutions to the QDC problem. We perform extensive experimental studies on a variety of real and synthetic networks to evaluate the performance of the proposed method. The results show that our method achieves much higher accuracy compared to the state-of-the-art methods and runs efficiently.

2. RELATED WORK

Global graph partitioning: Global graph partitioning has been extensively studied [11]. Representative methods include modularity clustering [27] and multi-level graph partitioning [17]. Some existing methods use edge weighting schemes to enhance the performance. For example, one method [18] strengthens the intracluster edges by rewarding edges with high common neighbor ratio, and weakens the intercluster edges by penalizing edges with high betweenness centrality. Other methods reward edges that have short cycles connecting their endpoints [4] or edges whose endpoints have similar degrees [6].

Local community detection: Given a set of query nodes, the local community detection problem aims at finding the community near the query nodes. Most existing methods optimize a goodness metric. The classic density definition [29], edge-surplus [36], and minimum degree [31, 9] maximize the internal denseness of the community. The subgraph modularity [23], density-isolation [22], and external conductance [33, 2, 19] try to maximize the internal denseness and minimize the external sparseness at the same time. The local modularity metric [7] optimizes the sharpness of the boundary of the community. Other methods enumerate different types of communities, such as the α-adjacency-γ-quasi-k-clique [8] and the k-truss community [14]. The α-adjacency-γ-quasi-k-clique is based on the γ-quasi-k-clique, which contains k vertices and at least $\lfloor \gamma \frac{k(k-1)}{2} \rfloor$ edges. The k-truss community is based on the k-truss subgraph, where every edge is contained in at least $(k-2)$ triangles within the subgraph.

3. THE FREE RIDER EFFECT

In this section, we first review some representative goodness metrics for local communities, and then discuss the free rider effect caused by them. Table 1 lists the main symbols used in this paper and their definitions.

3.1 Representative Goodness Metrics

Let $G(V, E)$ be an undirected graph and $G[S]$ be the subgraph induced by a set of nodes $S \subseteq V$. Let $f(S)$ be a goodness metric defined on subgraph $G[S]$. Most of the existing methods on local community detection can fit in the following generic formulation [7, 2, 23, 29, 31, 24, 36].

Table 1: Main symbols

Symbols	Definitions
$G(V, E)$	undirected graph G with node set V and edge set E
$Q; q$	a set of query nodes Q; a query node q
S; $\|S\|$	a set of nodes $S \subseteq V$; number of nodes in S
S^*	node set of the global optimal subgraph
S_l^*	node set of a local optimal subgraph
$G[S]$	subgraph induced by S
$f(S)$	a goodness metric defined on subgraph $G[S]$
$w(u, v)$	weight of an edge (u, v)
$e(S)$	sum of edge weights, $e(S) = \frac{1}{2}\sum_{u,v\in S} w(u, v)$
$e(S, T)$	sum of edge weights, $e(S, T) = \sum_{u\in S, v\in T} w(u, v)$
c	decay factor in the penalized hitting probability
$r(u)$	penalized hitting probability of u w.r.t. the query
$r(S)$	sum of proximity values, $r(S) = \sum_{u\in S} r(u)$
$\pi(u)$	query biased node weight, $\pi(u) = 1/r(u)$
$\pi(S)$	sum of query biased node weights, $\pi(S) = \sum_{u\in S} \pi(u)$
$\rho(S)$	query biased density, $\rho(S) = e(S)/\pi(S)$
N_u	the set of neighbor nodes of node u in graph G
$w_S(u)$	degree of u in $G[S]$, $w_S(u) = \sum_{v\in N_u\cap S} w(u, v)$
$w'_S(u)$	query biased degree in $G[S]$, $w'_S(u) = w_S(u)/\pi(u)$
$w(u); w'(u)$	degree of u in G; query biased degree of u in G
$\phi(S)$	volume of S, $\phi(S) = \sum_{u\in S} w(u)$
$\overline{S}$	complement of S, $\overline{S} = V \setminus S$
δS	boundary of S, $\delta S = \{u \in S \mid \exists v \in N_u \cap \overline{S}\}$

PROBLEM 1. [Local Community Detection] *Given an undirected graph $G(V, E)$, a set of query nodes Q, and a goodness metric $f(S)$, find the subgraph $G[S]$ such that*
1) $Q \subseteq S$;
2) $f(S)$ is optimized.

Table 2 lists some representative goodness metrics and their formulas. The classic density definition [29], edge-surplus [36], and minimum degree [31] measure the internal denseness of the local community. In the edge-surplus metric, h is a strictly-increasing function, and α is a constant. As proposed in [36], usually $h(x)$ is concave, or $h(x) = \binom{x}{2}$.

The subgraph modularity [23], density-isolation [22], and external conductance [2] measure both the internal denseness and external sparseness of the community. Note that α and β in the density-isolation formula are user specified constants [22]. In the external conductance metric [2], we have that $\phi(S) = 2e(S) + e(S, \overline{S})$. The total internal edge weight $e(S)$ measures the internal denseness, and the total external edge weight $e(S, \overline{S})$ measures the external sparseness.

The local modularity metric [7] measures the sharpness of the boundary of the community. In this metric, the numerator represents the total weight of the internal edges incident to δS, and the denominator represents the total weight of all edges incident to δS.

Note that among the metrics discussed above, the external conductance needs to be minimized, while other metrics need to be maximized.

3.2 Global Free Rider Effect

In this section, we discuss the global free rider effect. In this case, for any subgraph, its goodness value will increase if the global optimal subgraph is merged into it.

DEFINITION 1. [Global Optimal Subgraph] *The global optimal subgraph is the subgraph $G[S^*]$ whose goodness value $f(S^*) \geq f(S)$, for any $S \subseteq V$.*

Table 2: Goodness metrics and their free rider effects

Goodness metrics	Ref.	Formulas $f(S)$		Glo.	Loc.
Classic density	[29]	$e(S)/\|S\|$		✓	✓
Edge-surplus	[36]	$e(S) - \alpha h(\|S\|)$	concave $h(x)$	✓	✓
			$h(x) = \binom{x}{2}$	×	✓
Minimum degree	[31]	$\min_{u\in S} w_S(u)$		✓	✓
Subgraph modularity	[23]	$e(S)/e(S, \overline{S})$		✓	✓
Density-isolation	[22]	$e(S) - \alpha e(S, \overline{S}) - \beta\|S\|$		✓	✓
External conductance	[2]	$e(S, \overline{S})/\min\{\phi(S), \phi(\overline{S})\}$		×	✓
Local modularity	[7]	$e(\delta S, S)/e(\delta S, V)$		×	✓

A goodness metric f causes the global free rider effect if for any $S \subseteq V$, $f(S) \leq f(S \cup S^*)$. In this case, the optimal solution to Problem 1 must contain the global optimal subgraph $G[S^*]$.

In the next, we formally prove that a set of goodness metrics cause the global free rider effect. First, we review some definitions [30]: f is *submodular* if $f(S) + f(T) \geq f(S \cap T) + f(S \cup T)$, *supermodular* if $f(S) + f(T) \leq f(S \cap T) + f(S \cup T)$, and *modular* if it is both submodular and supermodular.

For example, $f(S) = e(S, \overline{S})$ is submodular; $f(S) = e(S)$ is supermodular; $f(S) = |S|$ is modular. The edge-surplus metric with a concave $h(x)$ and the density-isolation metric are both supermodular.

THEOREM 1. *If the goodness metric f is supermodular, then for any $S \subseteq V$, $f(S) \leq f(S \cup S^*)$.*

PROOF. Since f is supermodular, we have that $f(S) + f(S^*) \leq f(S \cap S^*) + f(S \cup S^*)$. Since $G[S^*]$ is the global optimal subgraph, we have that $f(S^*) \geq f(S \cap S^*)$. Combining these two inequalities, we have that $f(S) \leq f(S \cup S^*)$. □

Since the edge-surplus metric with a concave $h(x)$ and the density-isolation metric are both supermodular, we have the following lemma.

LEMMA 1. *The edge-surplus metric with a concave $h(x)$ and the density-isolation metric satisfy that for any $S \subseteq V$, $f(S) \leq f(S \cup S^*)$.*

We say that f is *monotonically increasing*, if for any $S \subseteq V$ and $S' \subseteq \overline{S}$, $f(S \cup S') \geq f(S)$.

THEOREM 2. *If $f(S) = \frac{g(S)}{h(S)}$, where $g(S) > 0$ is supermodular and monotonically increasing, and $h(S) > 0$ is submodular, then for any $S \subseteq V$, $f(S) \leq f(S \cup S^*)$.*

PROOF. Let $f'(S, \lambda) = g(S) - \lambda h(S)$, where λ is a real-valued constant. Let $\lambda = f(S)$. Proving that $f(S \cup S^*) \geq f(S)$ is equivalent to proving that $f'(S \cup S^*, \lambda) \geq 0$. Note that $g(S) - \lambda h(S) = 0$ since $\lambda = f(S)$. We have that

$$\begin{aligned} &f'(S \cup S^*, \lambda) = g(S \cup S^*) - \lambda h(S \cup S^*) \\ &\geq g(S) + g(S^*) - g(S \cap S^*) - \lambda(h(S) + h(S^*) - h(S \cap S^*)) \\ &= g(S^*) - g(S \cap S^*) - \lambda(h(S^*) - h(S \cap S^*)). \end{aligned}$$

Since g is monotonically increasing, $g(S^*) - g(S \cap S^*) \geq 0$. If $h(S^*) - h(S \cap S^*) \leq 0$, we have that $f'(S \cup S^*, \lambda) \geq 0$. Now suppose that $h(S^*) - h(S \cap S^*) > 0$. Since $G[S^*]$ is the global optimal subgraph, we have that $f(S^*) \geq f(S \cap S^*)$. Thus we can derive that

$$\frac{g(S^*) - g(S \cap S^*)}{h(S^*) - h(S \cap S^*)} \geq \frac{g(S^*)}{h(S^*)} = f(S^*) \geq \lambda.$$

So, we have that $f'(S \cup S^*, \lambda) \geq 0$, i.e., $f(S \cup S^*) \geq f(S)$. □

The classic density definition and subgraph modularity both have the ratio form as shown in Theorem 2. Thus we have the following lemma.

LEMMA 2. *The classic density definition and subgraph modularity satisfy that for any $S \subseteq V$, $f(S) \leq f(S \cup S^*)$.*

LEMMA 3. *The minimum degree metric satisfies that for any $S \subseteq V$, $f(S) \leq f(S \cup S^*)$.*

PROOF. If we merge $G[S^*]$ into $G[S]$, its minimum degree will not decrease. □

In summary, the classic density, edge-surplus with a concave $h(x)$, density-isolation, subgraph modularity, and minimum degree metrics all cause the global free rider effect.

3.3 Local Free Rider Effect

In this section, we discuss the free rider effects caused by the local optimal subgraph.

DEFINITION 2. [Local Optimal Subgraph] *A local optimal subgraph is the subgraph $G[S_l^*]$ whose goodness value $f(S_l^*) \geq f(S)$, for any $S \subseteq S_l^*$.*

That is, deleting any node(s) from a local optimal subgraph will decrease its goodness value. Note that by definition, the global optimal subgraph is also a local optimal subgraph.

Let $G[S]$ denote the target local community that contains the query nodes. A local optimal subgraph $G[S_l^*]$ could be irrelevant to the query nodes. A goodness metric f causes the local free rider effect if $f(S) \leq f(S \cup S_l^*)$ for an irrelevant local optimal subgraph $G[S_l^*]$. We can generalize Theorems 1 and 2 to the local optimal subgraphs.

THEOREM 3. *If the goodness metric f is supermodular, then for any $S \subseteq V$, $f(S) \leq f(S \cup S_l^*)$.*

THEOREM 4. *Suppose that $f(S) = \frac{g(S)}{h(S)}$, where $g(S) > 0$ is supermodular and monotonically increasing, and $h(S) > 0$ is submodular. If $f(S_l^*) \geq f(S)$, the goodness metric f satisfies that $f(S) \leq f(S \cup S_l^*)$.*

The proofs of Theorems 3 and 4 are similar to those of Theorems 1 and 2 respectively. Based on Theorems 3 and 4, we have the following lemmas.

LEMMA 4. *The edge-surplus metric with a concave $h(x)$ and the density-isolation metric satisfy that for any $S \subseteq V$, $f(S) \leq f(S \cup S_l^*)$.*

LEMMA 5. *If $f(S_l^*) \geq f(S)$, the classic density definition and the subgraph modularity metric satisfy that $f(S) \leq f(S \cup S_l^*)$.*

Next, we identify conditions that will cause the local free rider effects for the remaining goodness metrics.

LEMMA 6. *If $f(S_l^*) \geq f(S)$, the minimum degree metric satisfies that $f(S) \leq f(S \cup S_l^*)$.*

The following lemma says that if the target community and the irrelevant local optimal subgraph are node disjoint, and the goodness value of the irrelevant subgraph is large, then the edge-surplus metric with $h(x) = \binom{x}{2}$ causes the local free rider effect.

LEMMA 7. *If $S \cap S_l^* = \emptyset$ and $f(S_l^*) \geq \alpha \cdot |S| \cdot |S_l^*|$, where α is used in the definition of edge-surplus, then the edge-surplus with $h(x) = \binom{x}{2}$ satisfies that $f(S) \leq f(S \cup S_l^*)$.*

The following lemma says that if any pair of nodes u and v, where u is in the target community and v is in the irrelevant local optimal subgraph, are at least two hops away from each other, and the local optimal subgraph has a larger goodness value, then the local modularity metric causes the local free rider effect.

LEMMA 8. *If $S_l^* \cap (S \cup \delta\overline{S}) = \emptyset$ and $f(S_l^*) \geq f(S)$, the local modularity metric satisfies that $f(S) \leq f(S \cup S_l^*)$.*

Different from other metrics, the external conductance value needs to be minimized. In this case, a local optimal subgraph is the subgraph $G[S_l^*]$ whose goodness value $f(S_l^*) \leq f(S)$, for any $S \subseteq S_l^*$. A goodness metric f causes the local free rider effect if $f(S) \geq f(S \cup S_l^*)$ for an irrelevant local optimal subgraph $G[S_l^*]$. The following lemma says that if the volumes of the target community and the irrelevant local optimal subgraph are both small, and the irrelevant subgraph has a smaller goodness value, the external conductance metric causes the local free rider effect.

LEMMA 9. *If $\phi(S \cup S_l^*) \leq \phi(V)/2$ and $f(S_l^*) \leq f(S)$, the external conductance metric satisfies that $f(S) \geq f(S \cup S_l^*)$.*

In real-world networks, we can often find local optimal subgraphs that are irrelevant to the query nodes and satisfy the conditions discussed above. These irrelevant subgraphs will reduce the accuracy of the local community detection methods. Table 2 lists whether the goodness metrics cause the global or local free rider effect in the last two columns.

4. NODE WEIGHTING

In this section, we first use the random walk based proximity values to weight the nodes, and then define a new goodness metric, the query biased density. We show that after node weighting, the query biased densest subgraph is in the neighborhood of the query nodes.

4.1 Node Weighting by Random Walk

We first compute the proximity value of each node with regard to the query nodes. The reciprocal of the proximity value is used as the node weight. Thus the nodes closer to the query nodes will have smaller weights.

Many proximity measures can be used, such as random walk with restart [35] and the degree normalized penalized hitting probability [37]. In this paper, we use a variant of the degree normalized penalized hitting probability, which is referred to as the penalized hitting probability.

Let $w(u, v)$ be the edge weight between u and v, $w(u)$ be the degree of node u, and $w_{\max}$ be the maximum degree. The transition probability from u to v is $w(u, v)/w_{\max}$, which is normalized by the maximum degree. The penalized hitting probability penalizes the random walk for each additional step. The probability of hitting the query nodes for the first time is used as the proximity value. Let $r(u)$ denote the proximity value with regard to the query nodes Q. The penalized hitting probability can be defined as follows

$$r(u) = \begin{cases} 1, & \text{if } u \in Q; \\ c \sum_{v \in N_u} \frac{w(u,v)}{w_{\max}} \cdot r(v), & \text{if } u \in V \setminus Q, \end{cases}$$

where c ($0 < c < 1$) is the decay factor. The power iteration method can solve this linear system in $O(\kappa m)$ time, where κ is the number of iterations [28].

The query biased node weight $\pi(u)$ of node u is defined as the reciprocal of $r(u)$, i.e., $\pi(u) = 1/r(u)$. We always have that $0 \leq r(u) \leq 1$ and $\pi(u) \geq 1$.

Consider the example in Figure 1. The nodes in community A are densely connected to the query node through multiple short paths. Thus the random walker will have high probability to hit the query node starting from any node in A. On the other hand, there are only a few long paths connecting the query node and the nodes not in A. Starting from these nodes, the random walker will have low probabilities to hit the query node, since the probabilities are penalized by the path lengths. Thus the nodes in A will have higher proximity values than the nodes not in A. The distribution of the node weights, i.e., the reciprocal of the proximity values, are shown in Figure 3. A lighter color indicates a higher proximity value.

4.2 The Query Biased Density

Based on the query biased node weights, the query biased density is defined as follows.

DEFINITION 3. [Query Biased Density]

$$\rho(S) = \frac{e(S)}{\pi(S)},$$

where $\pi(S) = \sum_{u \in S} \pi(u)$ *is the sum of the query biased weights of nodes in* S.

If the node weights $\pi(u)=1$, the query biased density degenerates to the classic density $e(S)/|S|$. In the next, we will use the query biased density and density interchangeably if there is no ambiguity.

After node weighting, the densest subgraph is shifted to the neighborhood of the query nodes. For example, in the graph shown in Figure 1, before node weighting, the densest subgraph is B. Figure 3 shows the node weights after applying our node weighting scheme. A darker color represents a larger node weight. After node weighting, subgraph A becomes the query biased densest subgraph.

In the next, we show why the query biased densest subgraph will shift to the neighborhood of the query nodes. In particular, we prove that the query biased densest subgraph always (1) contains the query node, and (2) is connected, if the decay factor is small.

Let $|N_u|$ be the number of neighbors of u, and $N_{\max}$ be the maximum number of neighbors among all the nodes. The following lemma says that for any non-query node u, there exists a neighbor node v whose weight is smaller than that of node u times $w(u,v)/w(u)$.

LEMMA 10. *If* $c < (N_{\max})^{-1}$, *for any node* $u \in V \setminus Q$, *there exists a neighbor node* v *of node* u *(i.e.,* $v \in N_u$*), such that* $\pi(v) < \frac{w(u,v)}{w(u)} \cdot \pi(u)$.

PROOF. It is equivalent to prove that $w(u,v) \cdot r(v) > w(u) \cdot r(u)$. Suppose that node u violates this lemma, i.e., $\forall v \in N_u$, $w(u,v) \cdot r(v) \leq w(u) \cdot r(u)$. We have that $r(u) = c \sum_{v \in N_u} \frac{w(u,v)}{w_{\max}} r(v) \leq c \sum_{v \in N_u} \frac{w(u)}{w_{\max}} r(u) \leq c \cdot |N_u| \cdot r(u) < r(u)$. We get a contradiction that $r(u) < r(u)$. □

Based on this lemma, we can prove that if node u belongs to the query biased densest subgraph, the neighbor node v with small node weight also belongs to the query biased densest subgraph. Next, we show that the query biased densest subgraph is connected and contains the query node.

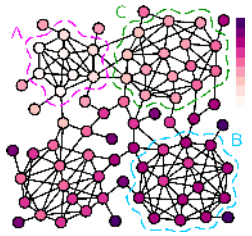


Figure 3: Effect of node weighting with $c = 0.9$ (darker color represents higher node weight; subgraph A is the query biased densest subgraph)

THEOREM 5. *If* $c < (N_{\max})^{-1}$ *and there is one query node, the query biased densest subgraph is connected and contains the query node.*

PROOF. Let $G[T]$ denote one connected component of the densest subgraph $G[S]$. We have that $\rho(T) \leq \rho(S)$. We also have that $\rho(T) \geq \rho(S)$, since otherwise $\rho(S \setminus T) > \rho(S)$. Thus $G[T]$ has equal density with $G[S]$. Suppose that $G[T]$ does not contain the query node.

First, let $u \in T$ be the node with the smallest value $\frac{\pi(u)}{w(u)}$, i.e., $\forall v \in N_u \cap T$, $\frac{\pi(v)}{w(v)} \geq \frac{\pi(u)}{w(u)}$. Since $w(v) \geq w(u,v)$, we have that $\forall v \in N_u \cap T, \pi(v) \geq \frac{w(u,v)}{w(u)} \cdot \pi(u)$. There exists a node $y \in N_u \cap \overline{T}$ such that $\pi(y) < \frac{w(u,y)}{w(u)} \cdot \pi(u)$. Otherwise, node u violates Lemma 10.

Next, we will prove that $\rho(T \cup \{y\}) > \rho(T)$. We have that $\frac{w(u)}{\pi(u)} \geq \frac{e(T)}{\pi(T)}$ because otherwise $\rho(T \setminus \{u\}) > \rho(T)$. Since $\pi(y) < \frac{w(u,y)}{w(u)} \cdot \pi(u)$, we have that $\frac{w(u,y)}{\pi(y)} > \frac{w(u)}{\pi(u)} \geq \frac{e(T)}{\pi(T)}$. So, we have that $\rho(T \cup \{y\}) \geq \frac{e(T)+w(u,y)}{\pi(T)+\pi(y)} > \rho(T)$.

In conclusion, if $G[T]$ does not contain the query node, there must exist a node in $\delta\overline{T}$, whose addition increases the density. This contradicts the assumption that $G[T]$ has the largest density. This completes the proof. □

Theorem 5 can be generalized to multiple query nodes.

THEOREM 6. *If* $c < (N_{\max})^{-1}$ *and there are multiple query nodes, each connected component of the query biased densest subgraph contains at least one query node.*

Please see the Appendix [1] for the proof.

4.3 Intuition behind the Query Biased Density

In this subsection, we further discuss the intuition on why the query biased density can help to reduce the free rider effect. The key observation is that the local community should not include the nodes that are well separated from it.

We examine two different ways to model the separateness between the local community and the irrelevant nodes. One model utilizes the conductance. As discussed before, the conductance of a subgraph can measure how well it is separated from the remaining graph. In this model, we assume that the local community has low conductance and show that the irrelevant nodes are unlikely to be included in the result using the query biased density. In another model, we assume that the local community is separated from the remaining graph by a set of low degree nodes. We show that in this case, irrelevant local optimal subgraphs are guaranteed

to be not included in the result. These theoretical results help explain why the query biased density method can reduce the free rider effect.

We still use $G[S]$ to denote the target local community. Let $r(\overline{S})$ be the sum of the proximity values of the nodes outside $G[S]$. The following theorem says that the expected value of $r(\overline{S})$ is upper bounded by the conductance of $G[S]$.

THEOREM 7. *Suppose that $\phi(S) < \phi(V)/2$. If we randomly pick a node in $G[S]$ with a probability proportional to its degree and use it as the query node, the expected value of $r(\overline{S})$ is no greater than the conductance of $G[S]$ times $\frac{c}{(1-c)^2}$.*

Please see the Appendix [1] for the proof. Theorem 7 says that if the conductance of $G[S]$ is small, the sum of the proximity values of the nodes outside $G[S]$ is also expected to be small. Thus these nodes are unlikely to be included in the result as a free rider to $G[S]$ because of their large weights (reciprocals of their proximity values).

The next theorem is based on the second model, i.e., the target local community is separated from the remaining graph by a set of low degree nodes. We still use $G[S_l^*]$ to denote a local optimal subgraph.

THEOREM 8. *Suppose that (1) $G[S]$ and $G[S_l^*]$ are separated by a set of nodes L with low degree $w(v) < \frac{e(S)}{|S|}$ (for any node $v \in L$); and (2) for any pair of nodes $u \in S$ and $v \in L$, $\pi(u) < \pi(v)$. If $c < (N_{\max})^{-1}$, the query biased density satisfies that $f(S) \geq f(S \cup S_l^*)$.*

The theorem assumes the degree of the nodes in L is less than half of the average node degree in $G[S]$. This is a reasonable assumption to model a sparse region separating $G[S]$ and $G[S_l^*]$. The second assumption is that the node weight of any node $u \in S$ is smaller than the node weight of any node $v \in L$, i.e., $\pi(u) < \pi(v)$. This is also a reasonable assumption, because in Theorem 7, we already show that the nodes outside $G[S]$ usually have smaller proximity values than the nodes in $G[S]$. Once these two assumptions are satisfied, the query biased densest subgraph will not have the free rider effect, no matter how dense the local optimal subgraph is. Please see the Appendix [1] for the proof.

5. OUR PROBLEM FORMULATION

In this section, we introduce a new formulation for the local community detection problem, i.e., the query biased densest connected subgraph (QDC) problem. We further study two related problems QDC$'$ and QDC$''$ with fewer constraints. The QDC problem is NP-hard, while QDC$'$ and QDC$''$ can be solved in polynomial time. We can often obtain the optimal or approximate solution for the QDC problem by solving QDC$'$ and QDC$''$.

5.1 The QDC Problem

We use the following problem formulation for the local community detection. The problem aims at finding the query biased densest subgraph that contains the query nodes and is also connected.

PROBLEM 2. [Query Biased Densest Connected Subgraph (QDC)] *Given an undirected graph $G(V, E)$ and a set of query nodes Q, find the subgraph $G[S]$ such that*
1) $Q \subseteq S$;
2) $\rho(S)$ is maximized;
3) $G[S]$ is connected.

THEOREM 9. *The QDC problem is NP-hard.*

5.2 Two Related Problems

In this section, we study two related problems that have fewer constraints than the QDC problem. The solutions to these two problems can be used to obtain the optimal or approximate solution to the QDC problem.

PROBLEM 3. [QDC$'$] *Given an undirected graph $G(V, E)$ and a set of query nodes Q, find the subgraph $G[S]$ such that*
1) $Q \subseteq S$;
2) $\rho(S)$ is maximized.

PROBLEM 4. [QDC$''$] *Given an undirected graph $G(V, E)$, find the subgraph $G[S]$ such that $\rho(S)$ is maximized.*

Compared to the QDC problem, QDC$'$ removes the connectivity constraint, and QDC$''$ further removes the constraint that $Q \subseteq S$. It is easy to see that these three problems have the following relationships.

LEMMA 11. *Let $G[S'']$, $G[S']$, and $G[S]$ be the solutions to* QDC$''$, QDC$'$, *and* QDC *respectively. We have that*
1) $\rho(S'') \geq \rho(S') \geq \rho(S)$;
2) If $Q \subseteq S''$, $G[S'']$ is also the solution to QDC$'$;
3) If $G[S']$ is connected, $G[S']$ is also the solution to QDC.

LEMMA 12. *Suppose that $G[S']$ is the solution to* QDC$'$ *and it is disconnected. If $G[S']$ has a connected component $G[T]$ containing all the query nodes and at least one non-query node, then $G[T]$ is a $\frac{\pi(T)}{\pi(T)-\pi(Q)}$-approximation of the solution to the* QDC *problem.*

PROOF. Let $G[S]$ be the solution to QDC. $G[S' \setminus (T \setminus Q)]$ is a feasible solution to QDC$'$ thus has smaller density than $G[S']$ does. Since $\frac{e(S')-e(T)}{\pi(S')-(\pi(T)-\pi(Q))} \leq \rho(S' \setminus (T \setminus Q)) \leq \rho(S')$, $\frac{e(T)}{\pi(T)-\pi(Q)} \geq \rho(S') \geq \rho(S)$. Thus $\rho(S) \leq \frac{\pi(T)}{\pi(T)-\pi(Q)}\rho(T)$. □

For any $u \in \overline{Q}$, $\pi(u) > 1$. If there is one query node, we have that $\pi(Q) = 1$, $\pi(T) > 2$. Thus the approximation ratio is smaller than 2.

As will be discussed in the next section, QDC$'$ and QDC$''$ can be solved in polynomial time. Although QDC is NP-hard, based on Lemmas 11 and 12, the solutions to QDC$'$ and QDC$''$ can be used to obtain the optimal or approximate solution to the QDC problem.

The overall procedure for the QDC problem: Algorithm 1 outlines the overall procedure for solving QDC. We first compute the optimal solution to QDC$'$. If it is connected, it is also the solution to QDC. If it is disconnected but it has one connected component containing all the query nodes and at least one non-query node, this connected component is returned as an approximate solution to QDC. Otherwise, we apply the heuristic algorithms to find a solution to QDC.

6. ALGORITHMS

In this section, we describe the main components of the overall procedure outlined in Algorithm 1. We start with the algorithm for QDC$''$.

Algorithm 1: The overall procedure for the QDC problem

Input: $G(V, E)$, query nodes Q, decay factor c

1: Compute the node weights with the decay factor c;
2: Compute the optimal solution $G[S]$ to the QDC$'$ problem;
3: **if** $G[S]$ *is connected* **then return** $G[S]$; **break**;
4: **if** $G[S]$ *has a connected component* $G[T]$ *containing all the query nodes* Q *and at least one non-query node* **then**
5: **return** $G[T]$; **break**;
6: Apply the heuristic algorithms to find a solution $G[S]$ to QDC;
7: **return** $G[S]$;

6.1 Algorithm for the QDC$''$ Problem

The algorithm to find the optimal solution of the QDC$''$ problem consists of the following steps.

1. Find a density threshold d by applying the greedy node deletion algorithm [5, 3] on G;
2. Find the d-core of G using the value d from step 1;
3. Find the densest subgraph from the d-core by the parametric maximum flow algorithm [12].

The first two steps are used to reduce the size of the original graph. The optimal solution to the QDC$''$ problem is guaranteed to be retained in the pruned graph. The exact parametric maximum flow algorithm is then applied to find the optimal solution in the pruned graph.

In step 1, the greedy node deletion algorithm keeps deleting the node with the lowest query biased degree: $w'(u) = w(u)/\pi(u)$. The subgraph with the maximum query biased density during the node deletion process is returned as the approximate solution. This algorithm outputs a 2-approximation to the QDC$''$ problem. The proof can be extended from that in [3] and is omitted here. The density of the identified subgraph will be used as the threshold d.

In step 2, the d-core is the maximal subgraph of G with all query biased node degrees no less than d. Any subgraph in which every node's query biased degree is no less than d is part of the d-core. Let $G[S]$ be the optimal solution of the QDC$''$ problem. Since any node $u \in S$ has query biased degree $w'_S(u) \geq \rho(S)$, $G[S]$ is a subgraph of d-core if $d \leq \rho(S)$. Thus using the d value identified in the previous step, we guarantee that the optimal solution is retained in the d-core.

In step 3, we apply the parametric maximum flow algorithm to find the densest subgraph in the d-core from step 2.

Let n and m be the number of nodes and edges in the original graph G, and n' and m' be the number of nodes and edges in the d-core. The greedy node deletion algorithm runs in $O(m + n \log n)$ [5]; the d-core is computed in $O(m)$ [3]; the parametric maximum flow algorithm runs in $O(n'm' \log(n'^2/m'))$ [12]. Experimental results show that $n'(m')$ is orders of magnitude smaller than $n(m)$.

6.2 Algorithm for the QDC$'$ Problem

In this section, we develop an exact polynomial time algorithm for the QDC$'$ problem. The algorithm uses the following *subgraph contraction* operation.

Contracting a subgraph $G[P]$ of $G(V, E)$ into a supernode p results in a new graph $G'(V', E')$ with $V' = \overline{P} \cup \{p\}$. The supernode p has weight $r(P)$, i.e., the sum of query biased weights of all nodes in P. Other nodes keep their original weights. The edge set E' is constructed as follows.

Figure 4: Subgraph contraction **Figure 5: Articulation nodes**

Algorithm 2: The algorithm for the QDC$'$ problem

Input: $G(V, E)$, query nodes Q, node weights π

1: $P_0 \leftarrow Q$; $G_0 \leftarrow$ contract $G[P_0]$ into a supernode p_0; $i \leftarrow 0$;
2: **while** *true* **do**
3: Compute the query biased densest subgraph $G_i[S_i]$ in G_i;
4: **if** S_i *contains the supernode* p_i **then break**;
5: $P_{i+1} \leftarrow S_i \cup P_i$; $G_{i+1} \leftarrow$ contract $G[P_{i+1}]$ into p_{i+1}; $i \leftarrow i+1$;
6: **return** $G[S_i \setminus \{p_i\} \cup P_i]$;

1. Keep edge (u,v) and its original weight $w(u,v)$ if $(u,v) \in E$ and $u, v \in \overline{P}$;
2. Add an edge (u, p) with weight $e(\{u\}, P)$ if $u \in \delta\overline{P}$;
3. Add a self-loop edge (p, p) with weight $e(P)$ if $e(P) > 0$.

Subgraph contraction operation preserves the density. That is, for any $S \subseteq \overline{P}$, subgraphs $G[P \cup S]$ and $G'[\{p\} \cup S]$ have the same density, so do subgraphs $G[S]$ and $G'[S]$.

The procedure is outlined in Algorithm 2. Initially, the subgraph induced by the query nodes is contracted into a supernode. In each iteration in lines 3-5, we find the query biased densest subgraph $G_i[S_i]$ by solving the QDC$''$ problem in G_i. If $G_i[S_i]$ does not contain the supernode p_i, the optimal solution of the QDC$'$ problem must contain $G[S_i]$, since $G[S_i]$ is the optimal solution for the QDC$''$ problem in G_i and adding it will increase the density of the current subgraph $G[P_i]$. Thus we can contract $G[S_i \cup P_i]$ into a supernode and repeat this process until the query biased densest subgraph $G_i[S_i]$ in G_i contains the supernode.

Figure 4 shows an example. In the left figure, all nodes and edges have unit weights. The purple node is the query node. The densest subgraph is the 6-clique in the green curve with density 2.5. Since it does not contain the query node, we contract it together with the query node into a supernode. The densest subgraph with density 2.38 in the new graph is indicated by the purple curve, and it contains the supernode. Thus, it is the optimal subgraph and highlighted by the purple curve in both figures.

Algorithm 2 runs in $O(\kappa t)$, where κ is the number of iterations and t is the running time of solving the QDC$''$ problem. At least one node is newly contracted into the supernode in each iteration, thus $\kappa \leq n$.

6.3 Algorithm for the QDC Problem

In this section, we develop two heuristic algorithms for the QDC problem if solving QDC$'$ does not give the desired solution. Note that our experimental results show that the probability of getting an optimal or approximate solution of QDC is very high by using the polynomial time algorithms introduced in the previous two subsections. With more than 90% probability, we get the optimal solution of QDC by solving QDC$'$. With more than 5% probability, we get an approximate solution of QDC by solving QDC$'$ (approximation ratio is shown in Lemma 12). Only with less than 5% probability, we need to apply the following heuristics to find a solution of QDC.

Heuristic1: Greedy node deletion with connectivity constraint. The heuristic is outlined in Algorithm 3. It iteratively deletes a set of non-articulation nodes [34] which

Algorithm 3: Greedy node deletion with connectivity constraint

Input: $G(V, E)$, query nodes Q, node weights π, parameter η

1: $V_0 \leftarrow V$; $i \leftarrow 0$;
2: **while** *true* **do**
3: Compute the articulation nodes S in $G[V_i]$; $S \leftarrow V_i \setminus S \setminus Q$;
4: **if** $|S| = 0$ **then break**;
5: Select a set of nodes $T \subseteq S$ that can be deleted together and has low query biased degree $w'_{V_i}(u) \leq \eta \cdot \rho(V_i)$, $\forall u \in T$;
6: **if** $|T| = 0$ **then** $T \leftarrow \{u \,|\, u = \text{argmin}_{v \in S} w'_{V_i}(v)\}$;
7: $V_{i+1} \leftarrow V_i \setminus T$; $i \leftarrow i + 1$;
8: $j \leftarrow \text{argmax}_i \rho(V_i)$; **return** $G[V_j]$;

Algorithm 4: The maximum adjacency search algorithm

Input: $G(V, E)$, query nodes Q, node weights π, parameter K

1: Compute the Steiner tree of Q, and let S be its node set;
2: $V_0 \leftarrow S$; $i \leftarrow 0$;
3: **while** $|V_i| \leq K$ **do**
4: $u \leftarrow \text{argmax}_{v \in \delta\overline{V_i}} e(\{v\}, V_i)/\pi(v)$;
5: $V_{i+1} \leftarrow V_i \cup \{u\}$; $i \leftarrow i + 1$;
6: $j \leftarrow \text{argmax}_i \rho(V_i)$; **return** $G[V_j]$;

have low query biased degrees. Here, the non-articulation nodes are the nodes whose deletion will not disconnect the graph [34]. The subgraph with the largest density during the node deletion process is returned.

Suppose that we have a set of biconnected components of graph G. In line 5, if we select at most one non-articulation node from each biconnected component, the deletion of this set of nodes will not disconnect the graph. This is because the deletion of any non-articulation node in one biconnected component does not disconnect this biconnected component.

Parameter η controls the degree of the non-articulation nodes to be deleted. η is usually set between $0 \sim 2$. Since $2\rho(V_i)$ is the average node degree, there are about half of the nodes whose query biased degree is smaller than the threshold $2\rho(V_i)$. More nodes are deleted in each iteration when larger η value is used. If all non-articulation non-query nodes have query biased degree greater than $\eta \cdot \rho(V_i)$, the algorithm picks the node with the minimum degree to delete. The algorithm runs in $O(nm)$ for finding the articulation nodes and biconnected components by depth first search [34].

Figure 5 shows an example. The articulation nodes are represented by squares, and the non-articulations nodes are represented by circles. The density of the original graph is 2.05. Suppose that $\eta = 1$. Then, the low degree node threshold is 2.05. After deleting some low degree non-articulation nodes, we get the subgraph as shown on the right in Figure 5.

Heuristic2: The maximum adjacency search algorithm. The heuristic is outlined in Algorithm 4. Mehlhorn's algorithm [26] is used to compute the Steiner tree given the query nodes. Thus the query nodes become connected. When computing the Steiner tree, the edge weight is set to the reciprocal of the original edge weight.

Next, the algorithm begins a local search process. The Steiner tree is used as the initial subgraph. In each iteration (lines 4-5), the algorithm adds the node $u \in \delta\overline{V_i}$ with the maximum adjacency value $e(\{u\}, V_i)/\pi(u)$ to V_i. The subgraph with the maximum density during the local search process is returned. Parameter K is used to control the search space. When the number of nodes in V_i is greater than K, the algorithm will terminate.

The local search process needs at most K iterations. Let N_{avg} be the average number of neighbors. Then, at the ith iteration, it takes $O(i \cdot N_{\text{avg}})$ time to find the node with the maximum adjacency value in line 4. Thus the local search process runs in $O(\sum_{i=0}^{K} i \cdot N_{\text{avg}}) = O(K^2 N_{\text{avg}})$. The Steiner tree can be computed in $O(m + n \log n)$ [26].

Table 3: Statistics of the real networks

Datasets	Abbr.	#Nodes	#Edges	#Communities
Amazon	AZ	334,863	925,872	151,037
DBLP	DP	317,080	1,049,866	13,477
Youtube	YT	1,134,890	2,987,624	8,385
Orkut	OR	3,072,441	117,185,083	6,288,363
LiveJournal	LJ	3,997,962	34,681,189	287,512
Friendster	FS	65,608,366	1,806,067,135	957,154

7. PROBLEM VARIANTS

In this section, we extend the algorithm for QDC to solve two variations of the basic QDC problem: (1) finding overlapping local communities, and (2) finding multiple disjoint local communities.

Finding overlapping local communities: In real applications, the same set of query nodes may belong to multiple different overlapping communities. We can extend the algorithm for QDC to solve this problem: after finding one community, we remove the nodes except the query nodes in the community; the algorithm can then be applied to find the next community. The algorithm runs in $O(k\tau)$, where k is the number of local communities and τ is the running time for finding one community.

Finding multiple disjoint local communities: Given a set of query nodes Q, a subset of Q may belong to a community, while other subsets may belong to entirely different communities. These communities are disjoint with each other. Our algorithm can also be extended to solve this problem. For each query node, we find its local community. Then we select the community with the maximum density. If this community contains several query nodes, then they are classified into this community. We remove the nodes in this community from the original graph. By applying this procedure repeatedly, we can partition the query nodes and identify their local communities. The algorithm runs in $O(|Q|^2\tau)$, where τ is the running time for finding one community.

8. EXPERIMENTAL RESULTS

We perform extensive experiments to evaluate the effectiveness and efficiency of the proposed methods using a variety of real and synthetic datasets. All the programs are written in C++. All experiments are performed on a server with 32G memory, Intel Xeon 3.2GHz CPU, and Redhat OS.

8.1 Datasets and State-of-the-Art Methods

The statistics of the real networks used in the experiments are shown in Table 3. These datasets are provided with ground-truth community memberships and are publicly available at *http://snap.stanford.edu*.

We compare our QDC method with several state-of-the-art local community detection methods, which are summarized in Table 4. These methods are categorized into three classes. The first class optimizes the internal denseness. The second class optimizes both the internal denseness and external sparseness. The third class optimizes the sharpness of the community boundary.

Table 4: State-of-the-art methods used for comparison

Classes	Abbr.	Ref.	Key idea
Internal	DS	[29]	Densest subgraph with query constraint
	OQC	[36]	Optimal quasi-clique; edge-surplus
	MDG	[31]	Minimum degree
Internal & External	PRN	[2]	External conductance
	LS	[25]	Local spectral
	EMC	[10]	More internal edges than external edges
	SM	[23]	Subgraph modularity
Boundary	LM	[7]	Local modularity

The densest subgraph with query constraint (DS) method finds the densest subgraph containing the query nodes [29]. The optimal quasi-clique with query constraint (OQC) method maximizes the edge-surplus goodness metric and also has the query constraint [36]. The minimum degree with global search (MDG) method maximizes the minimum degree and also has the query, size and distance constraints [31].

PageRank-Nibble (PRN) minimizes the external conductance [2]. The local spectral (LS) method is a revised version of the classic spectral method, and it is biased to the query node [25]. We implement the Spielman-Teng linear equation solver [32] in the LS method. The external minimum cut (EMC) method defines a community as a subgraph whose internal edges are more than external edges [10], and designs a minimum cut based algorithm to find local communities. The subgraph modularity (SM) and local modularity (LM) methods both use a heuristic local search procedure but have different goodness metrics [23, 7].

8.2 Evaluation Criteria

We use three different types of criteria to evaluate the selected methods: F-score, a set of community goodness metrics, and consistency.

F-score: It measures the accuracy of the detected community with regard to the ground-truth community labels. Given the discovered community $G[S]$ and the ground-truth community $G[T]$, F-score is defined as

$$F(S,T) = 2 \cdot \frac{\text{prec}(S,T) \times \text{rec}(S,T)}{\text{prec}(S,T) + \text{rec}(S,T)},$$

where $\text{prec}(S,T) = \frac{|S \cap T|}{|S|}$ is the precision and $\text{rec}(S,T) = \frac{|S \cap T|}{|T|}$ is the recall.

Community goodness metrics: We use three goodness metrics: *density*, *separability*, and *cohesiveness* [38]. *Density* adopts the classic density definition. Subgraph modularity is used to measure *separability* [38]. *Cohesiveness* is defined as the minimum internal conductance,

$$\min_{S' \subset S} \frac{e(S', S \setminus S')}{\min\{\phi_S(S'), \phi_S(S \setminus S')\}},$$

where $\phi_S(S')$ is the volume of S' in $G[S]$ [15]. Intuitively, a good local community should have high density, high separability, and high cohesiveness.

Consistency: Intuitively, using different sets of nodes in the same community as the query nodes, we should find the same community. Let $G[S]$ be the detected community when the query is Q, and $G[S']$ be the detected community when the query is $Q' \subseteq S$. The consistency of an algorithm is defined as

$$1 - \sqrt{\frac{1}{\binom{|S|}{|Q|}} \sum_{Q' \subseteq S, |Q'|=|Q|} (F(S,S') - F_{\text{mean}})^2},$$

Table 5: F-scores on real networks

F-score	QDC	DS	OQC	MDG	PRN	LS	EMC	SM	LM
AZ	**0.83**	0.52	0.54	0.46	0.69	0.66	0.61	0.60	0.58
DP	0.46	0.31	0.33	0.32	**0.48**	0.42	0.34	0.36	0.37
YT	**0.43**	0.23	0.22	0.17	0.26	0.24	0.21	0.21	0.22
OR	**0.47**	0.15	0.16	0.13	0.21	0.17	0.19	0.16	0.18
LJ	**0.64**	0.48	0.47	0.40	0.52	0.51	0.47	0.48	0.49
FS	**0.32**	–	0.14	0.12	0.17	0.16	–	0.14	0.13
$\overline{F}$	**0.53**	0.30	0.31	0.27	0.39	0.36	0.33	0.33	0.33
$\overline{\text{Prec}}$	**0.67**	0.46	0.45	0.29	0.51	0.41	0.34	0.38	0.48
$\overline{\text{Rec}}$	**0.72**	0.61	0.58	0.69	0.67	0.64	0.66	0.63	0.59

where F_{mean} is the mean value of $F(S,S')$ over all the S' [24]. That is, the consistency is defined as one minus the standard deviation of F-scores of the identified communities using different sets of query nodes from the community. We randomly take $\min\{10^3, \binom{|S|}{|Q|}\}$ subsets $Q' \subseteq S$ as the queries to estimate the consistency.

8.3 Evaluation on Real Networks

8.3.1 *Effectiveness Evaluation*

We first evaluate the effectiveness of the selected methods on real networks. We randomly pick 1000 sets of query nodes with size ranging from 1 to 40. Each set of query nodes is picked from a random ground-truth community. The decay factor is set to 0.9 in all experiments unless stated otherwise.

Table 5 shows the F-scores of the selected methods on different datasets. It can be seen that the QDC method outperforms other methods on most datasets except the DP dataset. The ground-truth community of DP network is based on conferences and the authors in one conference may not be densely connected [38]. The QDC method significantly improves the accuracy of the density based methods compared to DS and OQC. The PRN method has the second best performance. QDC outperforms PRN for about 10% on most datasets in terms of the F-score. Note that LS is also biased to the query node. However, its threshold constraint scheme cannot eliminate the free rider effect effectively. Please see the Appendix [1] for more detailed discussions. The average F-score $\overline{F}$, precision $\overline{\text{Prec}}$, and recall $\overline{\text{Rec}}$ over all datasets are also shown in this table. We can observe that the existing methods have high recall but low precision. This may be caused by the free rider effect.

Figure 6 shows the three goodness metrics, density, separability, and cohesiveness, of the detected communities on the LJ network. These metrics are normalized so that their maximum values are 1. The QDC method has the best overall performance. It has high cohesiveness, high density, and high separability. DS, OQC, and MDG have high density, but low cohesiveness and separability. PRN, LS, EMC, SM, and LM have high separability but low cohesiveness and density. All the selected existing methods suffer from the free rider effect. This is why their cohesiveness scores are low. Similar results can be observed in other datasets.

Table 6 shows the consistency of the detected communities using different methods. We can see that the consistency value of QDC is about 30% to 60% higher than those of other methods. The free rider effect causes the low consistency of other methods. If the nodes in the irrelevant free rider subgraph are selected as the query nodes, these methods will detect irrelevant communities.

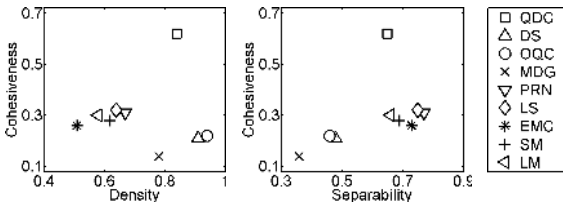


Figure 6: Goodness metrics on the LJ network

Table 6: Consistency on real networks

Consisten.	QDC	DS	OQC	MDG	PRN	LS	EMC	SM	LM
AZ	**0.94**	0.77	0.76	0.58	0.79	0.69	0.74	0.67	0.61
DP	**0.88**	0.62	0.64	0.37	0.65	0.53	0.56	0.43	0.56
YT	**0.85**	0.61	0.54	0.46	0.71	0.41	0.57	0.37	0.36
OR	**0.83**	0.56	0.52	0.32	0.68	0.43	0.51	0.54	0.47
LJ	**0.93**	0.74	0.67	0.43	0.84	0.64	0.73	0.58	0.52
FS	**0.78**	–	0.56	0.45	0.65	0.49	–	0.32	0.39
Average	**0.87**	0.64	0.62	0.44	0.72	0.53	0.61	0.49	0.49

Table 7: F-scores for different node weighting schemes

F-score	AZ	DP	YT	OR	LJ	FS	$\overline{F}$	$\overline{\text{Prec}}$	$\overline{\text{Rec}}$
PHP	0.83	0.46	0.43	0.47	0.64	0.32	0.53	0.65	0.78
RWR	0.73	0.39	0.33	0.35	0.56	0.28	0.45	0.52	0.69
PHP′	0.76	0.37	0.35	0.38	0.58	0.24	0.46	0.54	0.71

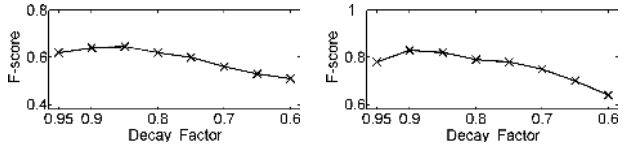


Figure 7: Tuning the decay factor (left: LJ; right: AZ)

In addition to penalized hitting probability (PHP), we also apply random walk with restart (RWR) [35] and degree normalized penalized hitting probability (PHP′) [37]. Table 7 shows the F-scores using different weighting schemes. We can see from the table that these three strategies provide similar performance, with PHP being slightly better. The differences between these three proximity measures are subtle. Please see the Appendix [1] for further discussions. Note that QDC with RWR or PHP′ still provides high F-scores compared to other state-of-the-art methods.

We further test the sensitivity of QDC with respect to the decay factor c. Figure 7 shows the F-scores on LJ and AZ datasets when varying the value of c from 0.95 to 0.6. It can be seen that the performance of QDC is stable for different values. When the decay factor is 0.9, the algorithm has the best performance. F-score slightly decreases when decreasing the decay factor.

8.3.2 *Efficiency Evaluation*

In this section, we first compare the overall running time of different methods, and then study the efficiency of each step of the QDC method.

Figure 8 shows the overall running time averaged over 1000 random queries. The QDC method can process large graphs with millions of nodes in tens of seconds. Note that even though some heuristic local search methods, such as LM and SM, run faster, their accuracies are low. The PRN and LS methods have similar performance, because both of them are based on the Spielman-Teng's method [32, 33]. The MDG method takes long running time because of the size and distance constraints in the problem formulation. The DS and EMC methods compute the maximum flow on the whole graph thus have long running time.

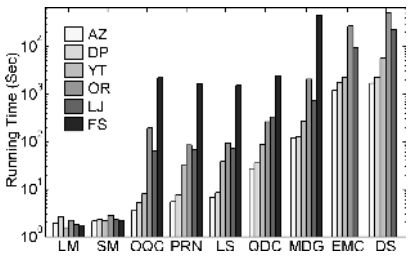


Figure 8: The overall running time of different methods

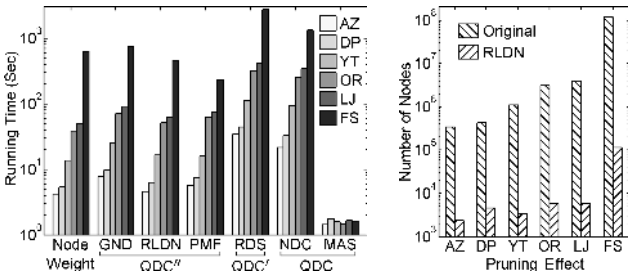


Figure 9: Left figure: running time of each step in QDC; right figure: pruning effect of the RLDN step

Recall that there are three major steps in solving QDC after node weighting. We first solve QDC′ exactly by repeatedly calling the algorithm for solving QDC″. If solving QDC′ does not give the optimal solution to QDC, then we try to find an approximate solution with approximation ratio shown in Lemma 12. If both previous steps do not give the solution to QDC, then we apply two heuristics, the greedy node deletion with connectivity constraint (NDC) method or the maximum adjacency search (MAS) method, to find a solution to QDC.

Among the 1000 random queries, we get the optimal solution of QDC by solving QDC′ for 915 queries. We get an approximate solution of QDC by solving QDC′ for 54 queries. We only need to apply heuristics NDC or MAS in 31 queries. Among the 915 queries, the query biased densest subgraph is the optimal solution to QDC for 813 queries.

The left figure in Figure 9 shows the running time of each step of the QDC method. Weighting a network with millions of nodes can be done in about 10 seconds. There are three steps in solving the QDC″ problem: greedy node deleting (GND) for finding the density threshold d, removing low degree node (RLDN) to find the d-core, and applying the parametric maximum flow (PMF) algorithm to find the densest subgraph. In solving QDC, heuristic MAS is more efficient than NDC, since NDC searches over the entire graph, while MAS only searches locally. Note that node weighting and solving QDC″ dominate the overall running time, since in most cases solving QDC″ will solve QDC.

The right figure in Figure 9 shows the pruning effect of the RLDN step in solving QDC″. The RLDN step reduces the number of nodes by 2 to 3 orders of magnitude. The parametric maximum flow step runs efficiently, since the RLDN step has significantly reduced the number of nodes.

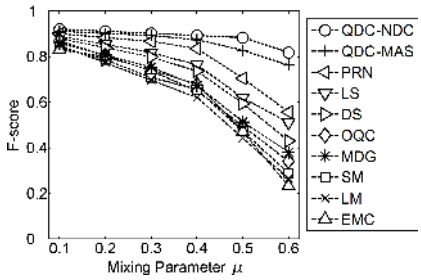


Figure 10: F-scores on the synthetic networks

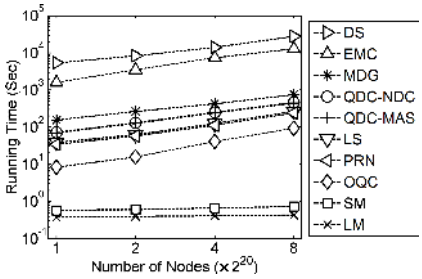


Figure 11: Running time on the synthetic networks

8.4 Evaluation on Synthetic Networks

We generate a collection of synthetic networks as proposed in [21] to evaluate the performance of the selected methods. The network generating model has three parameters γ, β, and μ. The degree and community size follow the power-law distribution with exponents γ and β respectively. We set $\gamma = 2$ and $\beta = 1$. The mixing parameter μ indicates the proportion of a vertex's neighbors that reside in other communities. By tuning μ, we can vary the clearness of the community structure: the boundaries between different communities become less clear for larger μ values.

We first evaluate the F-score when varying the mixing parameter μ from 0.1 to 0.6. The number of nodes in the network is 2^{20} and number of edges is 10^7. Figure 10 shows the F-scores. QDC-NDC (QDC-MAS) indicates that we use heuristic NDC (MAS). As we can see, the F-score decreases when increasing μ, i.e., lowering the clearness of the community structure. The QDC-NDC and QDC-MAS methods both have high F-scores, and clearly outperform other methods. Moreover, the QDC methods are very robust to the parameter μ with slight drop in accuracy for large μ values. The accuracies of all existing methods drop significantly after certain threshold. This is because when the boundary between the communities becomes less clear, it is easier for the existing methods to include irrelevant subgraphs in the identified communities due to the free rider effect.

We then evaluate the scalability when varying the graph size from 2^{20} to 2^{23} nodes. The edge density of each network is 9.53, and the mixing parameter $\mu = 0.3$. Figure 11 shows the running time. The local search methods SM and LM have almost constant running time. The OQC method uses a local search procedure but has increasing running time for larger networks. Other methods search the whole graph, thus the running time increases when increasing the graph size. The DS and EMC methods compute the maximum flow on the whole graph, thus they have long running time. The QDC method also computes the parametric maximum flow. Even so, the RLDN step in it significantly prunes the nodes. Thus the QDC method runs efficiently.

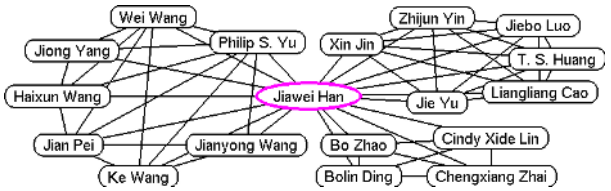
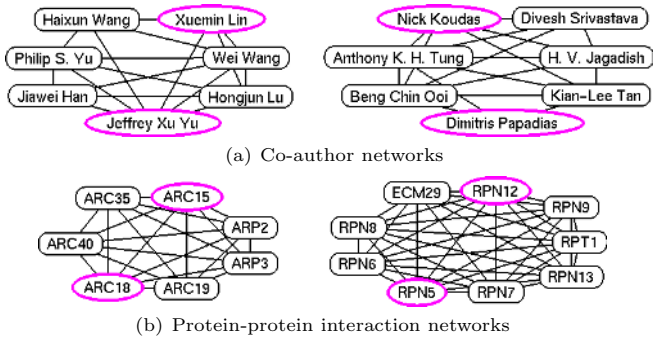


Figure 12: Three overlapping communities identified by QDC with Jiawei Han as the query author



(a) Co-author networks

(b) Protein-protein interaction networks

Figure 13: Finding multiple disjoint local communities

8.5 Case Study in Co-Author Networks and Biological Networks

Since the previous DBLP dataset does not include author names, we construct an author collaboration network from *http://dblp.uni-trier.de/xml/* for case study. A node in the network represents an author and the edge weight represents the number of papers that the two connected authors have co-authored. We remove the edges with weights less than 3. The network contains 236,497 nodes and 557,753 edges. There is no ground-truth community in this co-author network.

Figure 2(a) shows the communities detected by the existing DS method [29]. Clearly, the communities detected by DS have free riders. On the other hand, our QDC method only returns the communities in the upper part. Experimental results show that other methods also suffer from the free rider effect. The results are omitted due to space limitation.

We further evaluate the results of QDC on solving the two problem variants, i.e., finding overlapping local communities and finding multiple disjoint local communities.

Figure 12 shows three overlapping communities detected by QDC when using Jiawei Han as the query author. The authors in the left community are senior researchers in the core data mining research areas. The authors in the two communities on the right contain students and two senior researchers T. S. Huang and Chengxiang Zhai. The authors in the upper right community have published many works on social media mining. The authors in the lower right community mostly collaborate on information retrieval.

To find multiple disjoint local communities, in the co-author network, we use the set of authors {Jeffrey Xu Yu, Xuemin Lin, Dimitris Papadias, Nick Koudas} as the query authors. Figure 13(a) shows two detected communities. The left community is more about data mining, while the right one is more about database. Note that given a set of query nodes, no existing work can be directly applied to find multiple disjoint local communities.

We further apply QDC to protein-protein interaction networks to detect multiple protein complexes for a given set

of query proteins. We download the S. cerevisiae (yeast) protein-protein interaction network from BioGRID (*thebiogrid.org*). It contains 6,582 proteins and 224,724 unique physical bonding interactions. Many databases have curated the protein complexes, which can be used as the ground truth communities. We query the proteins {ARC15, ARC18, RPN5, RPN12}. Figure 13(b) shows the two detected communities. These two communities correspond to two protein complexes. The left community corresponds to the ARP2/3 complex, which contains the actin-related proteins. The right community is part of the 19/22s regulator complex, which is the proteasome regulatory particle [13].

9. CONCLUSIONS

Local community detection is a fundamental problem in network analysis and has attracted intensive research interests. Most existing local community detection methods optimize a goodness metric but suffer from the free rider effect. In this paper, we propose query biased node weighting, which shifts the query biased densest subgraph to the neighborhood of the query, to reduce the free rider effect. We study the QDC problem and two related problems, and develop efficient algorithms for them. Extensive experimental results demonstrate that the proposed method not only effectively reduces the free rider effect and achieves high accuracy, but also runs efficiently.

Acknowledgements. This work was partially supported by the National Science Foundation grants IIS-1162374, IIS-1218036, IIS-0953950, the NIH/NIGMS grant R01GM103309, and the OSC (Ohio Supercomputer Center) grant PGS0218. We would like to thank anonymous reviewers for their valuable comments.

10. REFERENCES

[1] http://filer.case.edu/yxw407 .
[2] R. Andersen, F. Chung, and K. Lang. Local graph partitioning using PageRank vectors. In *FOCS*, pages 475–486, 2006.
[3] B. Bahmani, R. Kumar, and S. Vassilvitskii. Densest subgraph in streaming and MapReduce. *PVLDB*, 5(5):454–465, 2012.
[4] J. W. Berry, B. Hendrickson, R. A. LaViolette, et al. Tolerating the community detection resolution limit with edge weighting. *Physical Review E*, 83(5):056119, 2011.
[5] M. Charikar. Greedy approximation algorithms for finding dense components in a graph. In *APPROX*, pages 139–152. 2000.
[6] M. Ciglan, M. Laclavík, and K. Nørvåg. On community detection in real-world networks and the importance of degree assortativity. In *KDD*, pages 1007–1015, 2013.
[7] A. Clauset. Finding local community structure in networks. *Physical Review E*, 72(2):026132, 2005.
[8] W. Cui, Y. Xiao, H. Wang, Y. Lu, and W. Wang. Online search of overlapping communities. In *SIGMOD*, pages 277–288, 2013.
[9] W. Cui, Y. Xiao, H. Wang, and W. Wang. Local search of communities in large graphs. In *SIGMOD*, pages 991–1002, 2014.
[10] G. W. Flake, S. Lawrence, and C. L. Giles. Efficient identification of web communities. In *KDD*, pages 150–160, 2000.
[11] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.
[12] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM J. Comput.*, 18(1):30–55, 1989.
[13] A.-C. Gavin et al. Proteome survey reveals modularity of the yeast cell machinery. *Nature*, 440(7084):631–636, 2006.
[14] X. Huang, H. Cheng, L. Qin, W. Tian, and J. X. Yu. Querying k-truss community in large and dynamic graphs. In *SIGMOD*, pages 1311–1322, 2014.
[15] R. Kannan, S. Vempala, and A. Vetta. On clusterings: good, bad and spectral. *JACM*, 51(3):497–515, 2004.
[16] R. M. Karp. *Reducibility among combinatorial problems*. Springer, 1972.
[17] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1998.
[18] A. Khadivi, A. A. Rad, and M. Hasler. Network community-detection enhancement by proper weighting. *Physical Review E*, 83(4):046104, 2011.
[19] K. Kloster and D. F. Gleich. Heat kernel based community detection. In *KDD*, pages 1386–1395, 2014.
[20] I. Kloumann and J. Kleinberg. Community membership identification from small seed sets. In *KDD*, pages 1366–1375, 2014.
[21] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4):046110, 2008.
[22] K. J. Lang and R. Andersen. Finding dense and isolated submarkets in a sponsored search spending graph. In *CIKM*, pages 613–622, 2007.
[23] F. Luo, J. Z. Wang, and E. Promislow. Exploring local community structures in large networks. *Web Intelligence and Agent Systems*, 6(4):387–400, 2008.
[24] L. Ma, H. Huang, Q. He, K. Chiew, J. Wu, and Y. Che. GMAC: a seed-insensitive approach to local community detection. In *DaWak*, pages 297–308, 2013.
[25] M. W. Mahoney, L. Orecchia, and N. K. Vishnoi. A local spectral method for graphs: with applications to improving graph partitions and exploring data graphs locally. *JMLR*, 13(1):2339–2365, 2012.
[26] K. Mehlhorn. A faster approximation algorithm for the Steiner problem in graphs. *IPL*, 27(3):125–128, 1988.
[27] M. E. Newman. Modularity and community structure in networks. *PNAS*, 103(23):8577–8582, 2006.
[28] Y. Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.
[29] B. Saha, A. Hoch, S. Khuller, L. Raschid, and X.-N. Zhang. Dense subgraphs with restrictions and applications to gene annotation graphs. In *RECOMB*, pages 456–472, 2010.
[30] A. Schrijver. *Combinatorial optimization: polyhedra and efficiency*. Springer, 2003.
[31] M. Sozio and A. Gionis. The community-search problem and how to plan a successful cocktail party. In *KDD*, pages 939–948, 2010.
[32] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC*, pages 81–90, 2004.
[33] D. A. Spielman and S.-H. Teng. A local clustering algorithm for massive graphs and its application to nearly-linear time graph partitioning. *SIAM J. Comput.*, 42(1):1–26, 2013.
[34] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.
[35] H. Tong, C. Faloutsos, and J. Pan. Fast random walk with restart and its applications. In *ICDM*, pages 613–622, 2006.
[36] C. Tsourakakis, F. Bonchi, A. Gionis, F. Gullo, and M. Tsiarli. Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In *KDD*, pages 104–112, 2013.
[37] Y. Wu, R. Jin, and X. Zhang. Fast and unified local search for random walk based k-nearest-neighbor query in large graphs. In *SIGMOD*, pages 1139–1150, 2014.
[38] J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. In *ICDM*, pages 745–754, 2012.