Supplemental Material of "Query-Driven Association Discovery in Large Networks"

1 Theorems and Proofs

Theorem 1 *Algorithm 1 (OPT-α) finds an α-association core containing $V(G_s) \cap Q$ with the largest α value.*

Proof. Indeed, the association strength of a vertex is node-monotonic non-increasing [Sozio and Gionis, 2010]. For a vertex v, its only changing factor of its association strength $as_{G_l,Q}(v,t) = \sum_{u \in N_{G_l}(v)} filter_{G_l,Q}(u,v) \cdot attr_{G_l,Q}(v,t)$ is $N_{G_l}(v)$. Since $N_{G_l}(v) \subseteq N_{G_s}(v)$ in any induced subgraph G_l of G_s, $as_{G_l,Q}(v,t) \subseteq as_{G_s,Q}(v,t)$.

Due to the node-monotonic non-increasing property of the association strength, iteratively removing the vertex with the minimum association strength can find an α-association core with the largest α value. The proof is inline with that of [Sozio and Gionis, 2010]. □

Theorem 2 *Algorithm 1 (OPT-α) runs in $O(|Q \cap V(G_s)|m_s + n_s m' + m_s log n_s)$ where $n_s = |V(G_s)|$, $m_s = |E(G_s)|$, and m' ($m' \leq m_s$) is the number of visited edges to check the connectivity of $Q \cap V(G_s)$ in a subgraph.*

Proof. According to Equation 3, computing the association strength for each vertex takes $O(|Q|m_s)$ using D and H. Checking the connectivity of $Q \cap V(G_s)$ in G_l can be done in $O(m')$. As only one vertex is deleted in each iteration, OPT-α iterates $O(n_s)$ times. Therefore, checking connectivity takes $O(n_s m')$ totally. If the binary heap is adopted, deleting u with the minimum association strength and adjusting the association strengths of u's neighbors can be done in $O(m_s log n_s)$. As a result, OPT-α runs in $O(|Q|m_s + n_s m' + m_s log n_s)$. □

Theorem 3 *Algorithm 2 (OPT-split) runs in $O(|Q'|m_s + n_s m'_s + m_s log n_s)$ where $Q' = V(G_s) \cap Q$, $m_s = |E(G_s)|$, $n_s = |V(G_s)|$, and m'_s is the number of visited edges to check the connectivity of the query nodes contained in each connected component ($\mathbb{G}_c$ in Line 17 of OPT-split).*

Proof. For a vertex, computing its preference difference takes $O(|Q'|)$. Hence computing the preference difference for each vertex and obtaining all the preference gaps can be done in $O(n_s|Q'|)$. Fetching the largest preference gap takes $O(|Q'|^2)$. Making a partition of G_s runs in $O(n_s|Q'|)$. Using the breadth-first search, $\mathbb{G}_c$ can be obtained in $O(m_s)$. As each subgraph in $\mathbb{G}_c$ contains at least one query node, $|\mathbb{G}_c| \leq |Q'|$. For each subgraph $G_i \in \mathbb{G}_c$, OPT-α takes $O(|Q \cap V(G_i)|m_i + n_i m'_i + m_i log n_i)$ where $m_i = |E(G_i)|$, $n_i = |V(G_i)|$, and m'_i is the number of traversed edges to check the connectivity of $Q \cap V(G_i)$ in G_i. Hence $\sum_{i=1}^{|\mathbb{G}_c|} |Q \cap V(G_i)|m_i + n_i m'_i + m_i log n_i \leq |Q'|m_s + n_s m'_s + m_s log n_s$. Therefore, Algorithm 2 can be done in $O(|Q'|m_s + n_s m'_s + m_s log n_s)$. □

Theorem 4 *Denote $G'(V,E)$ as the subgraph induced from $G(V,E)$ by $V_h = \{v | v \in V(G) \wedge \exists q \in Q, heat_{G,q}(v,t) > 0\}$ where the heat values are computed by hk-relax. Then OPT-QAD runs in $O(|Q|T(t,\varepsilon) + |Q|m_h + n_h m'_h + m_h log n_h)$ where $T(t,\varepsilon)$ is the time complexity of hk-relax, $m_h = |E(G')|$, $n_h = |V(G')|$ and m'_h is the number of visited edges to check the connectivity of the query nodes contained in each connected component of $G'(V,E)$.*

Proof. Computing the heat value of each vertex from each query node takes $O(|Q|T(t,\varepsilon))$ (Line 3). Using the vertices with positive heat values, inducing a subgraph $G'(V,E)$ from $G(V,E)$ runs in $O(m_h)$ (Line 4). The shortest distance from each vertex to each query node can be computed by $|Q|$ breadth-first search (Line 6), which takes $O(|Q|m_h)$.

Next, the loop of OPT-QAD (Lines 8-16) runs in $O(|Q|m_h + n_h m'_h + m_h log n_h)$. In each iteration, both OPT-α and OPT-split run in $O(|Q_c|m_c + n_c m'_c + m_c log n_c)$ where $Q_c = V(G_c) \cap Q$, $m_c = |E(G_c)|$, and $n_c = |V(G_c)|$ (Theorem 2 and Theorem 3). Only the subgraphs in $\mathbb{S}_c$ may join in the further computation. Hence the time complexity caused by a G_c during the loop is the following:

$$T(G_c) = O(Q_c m_c + n_c m'_c + m_c log n_c) + \sum_{i=1}^{|\mathbb{S}_c|} T(S_{ci}) \quad (1)$$

where S_{ci} is the ith element of $\mathbb{S}_c$ and $T(S_{ci})$ is the time complexity caused by S_{ci}. By expanding $T(S_{ci})$, Equation 1 can be rewritten as:

$$T(G_c) = O(Q_c m_c + n_c m'_c + m_c log n_c) + \sum_{i=1}^{|\mathbb{S}_c|} O(Q_{ci} m_{ci} + n_{ci} m'_{ci} + m_{ci} log n_{ci}) + \sum_{i=1}^{|\mathbb{S}_c|} \sum_{j=1}^{|\mathbb{S}_{ci}|} T(S_{cij}) \quad (2)$$

As $|Q_c| \geq \sum_{i=1}^{\mathbb{S}_c} |Q_{ci}|$, $m_c \geq \sum_{i=1}^{\mathbb{S}_c} |m_{ci}|$, and $n_c \geq$

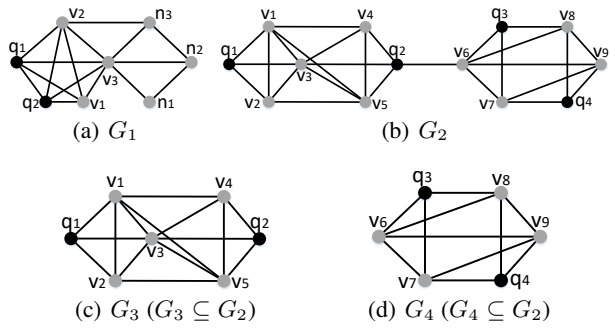


Figure 1: Examples for illustrating the proposed methods. In each subgraph, black circles are the query nodes.

$\sum_{i=1}^{\mathbb{S}_c} |n_{ci}|$, we have

$$T(G_c) \leq O(Q_c m_c + n_c m'_c + m_c log n_c) + \sum_{i=1}^{|\mathbb{S}_c|} \sum_{j=1}^{|\mathbb{S}_{ci}|} T(S_{cij}) \quad (3)$$

.

As $T(G_c)$ can be further expanded like Equation 2, the following equation is satisfied:

$$T(G_c) \leq O(Q_c m_c + n_c m'_c + m_c log n_c) + \sum_{G_r \in \mathbb{R}} T(G_r) \quad (4)$$

As $\sum_{G_r \in \mathbb{R}} T(G_r) = 0$, $T(G_c) \leq O(Q_c m_c + n_c m'_c + m_c log n_c)$. Similar to Equation 3, the loop (Lines 8-16) of Algorithm 3 (OPT-QAD) runs in $O(|Q| m_h + n_h m'_h + m_h log n_h)$.

Totally, OPT-QAD runs in $O(|Q| T(t, \varepsilon) + |Q| m_h + n_h m'_h + m_h log n_h)$. □

2 Running Examples

A running example of OPT-α. For instance, given the graph G_1 in Fig.1(a) with $Q = \{q_1, q_2\}$ and $t = 3$, the vertices n_1, n_2, n_3, v_1 and q_2 are removed sequentially. As Q is no longer connected when q_2 is removed, the loop is terminated. Thereafter, the subgraph G_s consisting of q_1, q_2, v_1, v_2 and v_3 with $min\{as_{G_s,Q}(v, t) | v \in V(G_s)\} = as_{G_s,Q}(q_1, t) = 12$ is returned.

A running example of OPT-QAD. Given the graph G_2 in Fig.1(b) with $Q = \{q_1, q_2, q_3, q_4\}$, $t = 3$, and $\beta = 2$, $\mathbb{G}_c = \{G_2\}$ is obtained after Lines 1-7 in Algorithm 3 (OPT-QAD). With G_2 fetched from $\mathbb{G}_c$, $R_\alpha = G_2$ is obtained applying OPT-α. Utilizing OPT-split, $\mathbb{S}_c = \mathbb{S}_\alpha = \{G_3, G_4\}$ is computed. As $\{G_3, G_4\}$ has larger effective association than $\{G_2\}$, $\mathbb{G}_c = \{G_3, G_4\}$. In the following iterations, since dealing G_3 or G_4 with OPT-split will result in 0-association-cores (a connected subgraph containing only one query node is a 0-association core), $\mathbb{R} = \{G_3, G_4\}$ is returned with $EA(\{G_3, G_4\}, Q, \beta, t) = 2^2 \cdot 3.1 + 2^2 \cdot 6.2 = 37.2$.

3 Parameter Evaluation

The parameters of OPT-QAD are evaluated in this section. t and ε are the parameters of heat kernel diffusion which affects the attraction of a vertex (For a vertex, its association strength is affected by its attraction according to Equation 3 in the submitted paper). β is the weighting factor of the effective association. 200 queries are generated for each network in Table 1 . In detail, each query is generated as follows: (1) Five communities are selected from 5,000 ground-truth communities. (2) In each selected community, three nodes are selected as the query nodes. (3) Five extra query nodes are selected from the network as outliers. All the communities and query nodes are selected using the drawn-by-drawn method (a simple sampling method). As a result, each query consists of 20 query nodes. The answer for the query is all the members from the selected communities.

Table 1: Network statistics(K=10^3 and M=10^6)

Network	Abbr.	$\|V\|$	$\|E\|$	Diameter
DBLP	DP	317K	1M	21
Youtube	YT	1.1M	3M	20
LiveJournal	LJ	4M	35M	17
Orkut	OR	3.1M	117M	9

The following criteria are used to measure the quality of the returned result:

$F_1 = \frac{2 \cdot |C \cap C_T|}{|C| + |C_T|}$ measures the similarity between a set of discovered members C and the set of members from the selected ground-truth communities C_T.

Given a graph $G(V, E)$ and a subgraph $S(V, E)$, the conductance of S is $\frac{c_S}{2 \cdot m_S + c_S}$ where $m_S = |E(S)|$ and $c_S = |\{(u, v) \in E(G) : u \in V(S), v \notin V(S)\}|$. If the conductance of S is small, S is well separated from G.

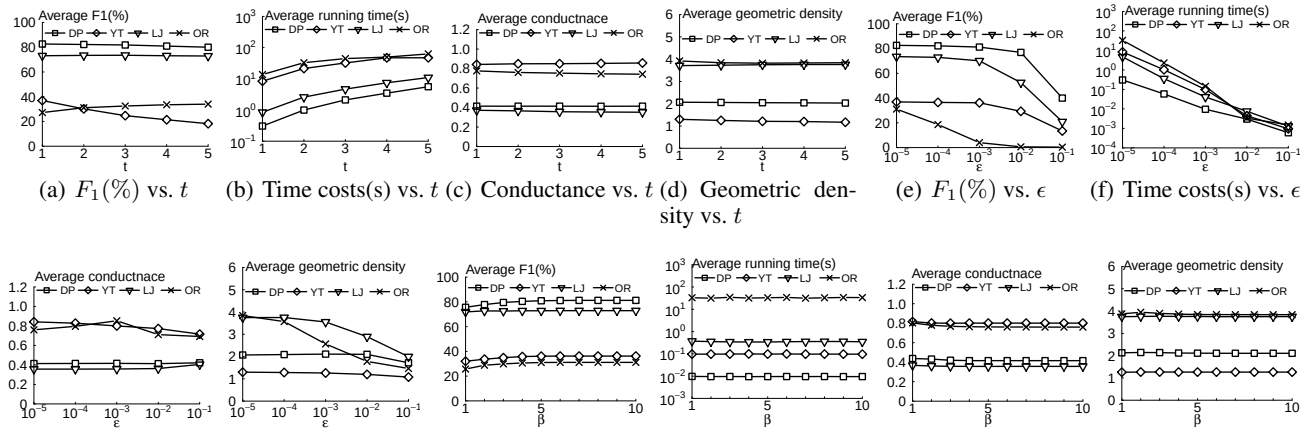
The geometric density $\frac{2 \cdot |E(S)|}{|V(S)| \cdot (|V(S)|-1)^{0.5}}$ of a subgraph $S(V, E)$ is the geometric mean of $\frac{|E(S)|}{|V(S)|}$ and $\frac{|E(S)|}{|V(S)| \cdot |V(S)-1|}$. A subgraph with large $\frac{|E(S)|}{|V(S)| \cdot |V(S)-1|}$ tends to be small in size, which cannot reveal rich associations among the query nodes. In addition, study [Wu *et al.*, 2015] shows that a subgraph with the large average degree may contain vertices not related to the query nodes. Hence the geometric density is used to measure the density of a subgraph while alleviating the drawbacks of the average degree and internal density.

In this experiment, the conductance and geometric density are used to measure the quality of the discovered subgraphs. As OPT-QAD may result in several connected subgraphs for a query, the conductance for a query is the average conductance of each subgraph and the geometric density for a query is the average geometric density of each subgraph.

Evaluating t. Fig.2(a)-(d) show the performance of OPT-QAD with t ranging from 1 to 5. ε is set to 10^{-5} and β is set to 8 in this experiment.

Fig.2(a) shows the average F_1. In different networks, OPT-QAD achieves the best F_1 with different t values. The performance of OPT-QAD is sensitive to t in Youtube and Orkut. As t gets larger, the association strength values of the vertices not close to the query nodes increase faster. Therefore, the performance of OPT-QAD in Youtube shows that vertices closer to the query nodes are better to reveal the associations among the query nodes. In Orkut, the ground-truth communities are large in size. In order to retrieve the members in the ground-truth communities, t should be larger to make the members have larger association strength values. Therefore, F_1 in Orkut increases with the increasing t.

Fig.2(b) presents the average time costs. As t increases,



(a) $F_1(\%)$ vs. t (b) Time costs(s) vs. t (c) Conductance vs. t (d) Geometric density vs. t (e) $F_1(\%)$ vs. ϵ (f) Time costs(s) vs. ϵ

(g) Conductance vs. ϵ (h) Geometric density vs. ϵ (i) $F_1(\%)$ vs. β (j) Time costs(s) vs. β (k) Conductance vs. β (l) Geometric density vs. β

Figure 2: Parameter evaluation of OPT-QAD

OPT-QAD becomes slower in each network. For one thing, a large t value requires OPT-QAD visit more vertices to approximate heat values. For another, with a large t value, the large amount of vertices having heat values increase the time costs of OPT-α and OPT-split in OPT-QAD.

Fig.2(c) and Fig.2(d) report the average conductance and the average geometric density respectively. Since the association strength considers the dense connectivity of a vertex to the query nodes, the subgraphs discovered by OPT-QAD are also densely connected. As a result, the conductance and geometric density vary little with t.

Evaluating ε. Fig.2(e)-(h) present the performance of OPT-QAD with ε ranging from 10^{-5} to 10^{-1}. In this experiment, β is set to 8. Besides, considering F_1 in Fig.2(a) and time costs in Fig.2(b), t is set to 1, 1, 3, and 2 for DBLP, Youtube, LiveJournal, and Orkut respectively.

Fig.2(e) reports the average F_1. F_1 decreases with the increasing ε. This is because some of the vertices associated with the query nodes are removed due to the large ε (which indicates a large error tolerance). Hence F_1 decreases.

Fig.2(f) shows the average running time. As ε increases, the efficiency of OPT-QAD is improved. The larger the ε is, the fewer vertices should be visited to approximate the heat values. Besides, with fewer vertices having heat values, OPT-α and OPT-split speed up. Therefore, OPT-QAD accelerates with increasing ε.

Fig.2(g) presents the average conductance. As ε increases, many vertices densely connected to the query nodes are removed. As a result, the conductance of DBLP and LiveJournal increases. In Orkut, when $\varepsilon \geq 10^{-3}$, the conductance decreases. This is because ground-truth communities in Orkut are large in size. With the increasing ε, many members will not have heat values. As a result, only small and densely connected subgraphs are returned, which decreases the conductance. In Youtube, the query nodes are far from each other and not densely connected in general, when the ε gets larger, only small and densely connected subgraphs can be returned. As a result, the conductance in Youtube decreases with the increasing t.

Fig.2(h) provides the average geometric density. As ε raises, many vertices which are densely connected to the query nodes are pruned. Hence the geometric density declines with the increasing ε in each network.

Evaluating β**.** Fig.2(i)-(l) report the performance of OPT-QAD with β ranging from 1 to 10. In this experiment, t is set to the same values as the ones in the experiment of evaluating ε. Considering the performance of OPT-QAD varying with ε, ε is set to 10^{-3}, 10^{-3}, 10^{-4}, and 10^{-5} for DP, YT, LJ, and OR respectively.

Fig.2(i) show the average F_1. With the increasing β, OPT-QAD tends to discover more weak associations among the query nodes. Hence F_1 raises as β increases. When β is large enough, weak associations can be effectively discovered. Therefore, F_1 becomes stable with large β.

Fig.2(j)-(l) present the average time costs, the average conductance and the average geometric density respectively. As OPT-QAD can effectively find the subgraphs in which the query nodes are well associated with each other, the metrics vary little with β.

4 More Comparison Results

In this section, a natural method called BASE is designed to compared with OPT-QAD. Given a graph $G(V, E)$, a set of query nodes Q, and a threshold c, BASE proceeds in a sequence of steps as follows:

1. For each query node $q \in Q$, a subgraph G_i is discovered using a method of community search. Denote $\mathbb{S}$ as the set of discovered subgraphs.
2. The subgraphs in $\mathbb{S}$ are merged iteratively by Jaccard similarity $J(S_i, S_j) = \frac{|V(S_i) \cap V(S_j)|}{|V(S_i) \cup V(S_j)|}$ where $S_i, S_j \in \mathbb{S}$:
   2.1. Find two subgraph G_i and G_j from $\mathbb{S}$ achieving the largest Jaccard similarity.
   2.2. If $J(G_i, G_j) \geq c$, $G_{i \cup j} = G_i \cup G_j$, $\mathbb{S} = \mathbb{S} \cup \{G_{i \cup j}\} \setminus \{G_i, G_j\}$; Otherwise, $\{G' | G' \in \mathbb{S} \wedge |V(G') \cap Q| > 1\}$ is returned.

LCTC (A state-of-the-art of community search) is applied in Step 1 due to its outstanding performance in discovering ground-truth communities [Huang *et al.*, 2015].

Table 2: Average $F_1(\%)$ of the comparing methods

Network	BASE-0.3	BASE-0.6	BASE-0.9	OPT-QAD
DP	66.7	64	62	**81.2**
YT	17.3	8.7	4.3	**36.2**
LJ	70	65.7	59.3	**72.9**
OR	7.1	0.9	0.2	**31.1**

With the same queries used in Section 5.2, we report results using 3 different c settings, 0.3, 0.6, and 0.9, for BASE. As BASE may also return multiple subgraphs, we remove the subgraphs containing a single query node for accuracy.

Table 2 reports the average F_1. Using the similarity of each community discovered by LCTC, BASE is effective to discover strong associations among query nodes. However, the members discovered for a single query node are very close to the node in BASE. As a consequence, BASE cannot discover associations among the query nodes which are associated with but not very close to each other. Therefore, it is not as effective as OPT-QAD.

5 More Case Studies

References

[Huang *et al.*, 2015] Xin Huang, Laks V.S. Lakshmanan, Jeffrey Xu Yu, and Hong Cheng. Approximate closest community search in networks. *Proceedings of the VLDB Endowment*, 9(4):276–287, 2015.

[Sozio and Gionis, 2010] Mauro Sozio and Aristides Gionis. The community-search problem and how to plan a successful cocktail party. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 939–948, 2010.

[Wu *et al.*, 2015] Yubao Wu, Ruoming Jin, Jing Li, and Xiang Zhang. Robust local community detection: on free rider effect and its elimination. *Proceedings of the VLDB Endowment*, 8(7):798–809, 2015.