2019 IEEE 35th International Conference on Data Engineering (ICDE)

Contextual Community Search over Large Social Networks

Lu Chen†, Chengfei Liu†, Kewen Liao‡, Jianxin Li¶, Rui Zhou†
†Swinburne University of Technology, ‡Charles Darwin University, ¶Deakin University
†{luchen, cliu, rzhou}@swin.edu.au, ‡Kewen.liao@cdu.edu.au ¶jianxin.li@deakin.edu.au

***Abstract*—Community search on attributed networks has recently attracted great deal of research interest. However, most of existing works require query users to specify some community structure parameters. This may not be always practical as sometimes a user does not have the knowledge and experience to decide the suitable parameters. In this paper, we propose a novel parameter-free contextual community model for attributed community search. The proposed model only requires a query context, i.e., a set of keywords describing the desired matching community context, while the community returned is both structure and attribute cohesive w.r.t. the provided query context. We theoretically show that both our exact and approximate contextual community search algorithms can be executed in worst case polynomial time. The exact algorithm is based on an elegant parametric maximum flow technique and the approximation algorithm that significantly improves the search efficiency is analyzed to have an approximation factor of $\frac{1}{3}$. In the experiment, we use six real networks with ground-truth communities to evaluate the effectiveness of our contextual community model. Experimental results demonstrate that the proposed model can find near ground-truth communities. We also test both our exact and approximate algorithms using eight large real networks to demonstrate the high efficiency of the proposed algorithms.**

I. Introduction

The prevalence of the internet has created large-scale collections of networked data, with online services encouraging social interaction, e.g., Facebook, Twitter, and fostering business communication, e.g., Email, LinkedIn. By studying the structures and attributes of large scale linked data, researchers have discovered that data often exhibits correlation among the attributes of linked individuals. Further work has considered the root cause of this correlation and distinguished between social influence, which is the tendency for linked individuals to adopt the characteristics of their neighbours, and homophily, where links are created based on the attribute similarities of the individuals [15]. On such attributed network data, discovering and analysing different community structure is a topic of great interest and importance for its economical and marketing implications such as online advertisement, online recommendation, and social tie/link prediction.

Prior works. While community search has attracted widespread studies, community modelling given target attributes as a context has not been considered seriously by researchers. The majority of research in this area has focused on pure structure-based community search without vertex attributes, e.g., k-core [23], k-truss [2] and k-edge-connected graph [28]. Existing works focus on local attributed community search for a given set of query vertices, in which the resultant communities must contain the query vertices and other included vertices should be highly relevant to the query vertices [5, 11]. Other works [1, 17, 27] focusing on global community search are not confined on a set of vertices but still require users to specify structure and/or attribute cohesiveness parameters of the desired communities. Due to the high complication of large scale attribute networks, it is sometimes difficult for query users to express suitable queries without prior knowledge. Therefore, it is highly desirable to come up with the mechanism of parameter-free attributed community search.

Our work. In this paper, we propose a novel parameter-free community model, namely the *contextual community* that only requires users to provide a set of keywords describing an application/user context (e.g. location and preference). This is different from the existing community models depending on structural parameters (e.g. k as the minimum vertex degree in k-core). It is also different from keyword subgraphs, which target the minimal subgraphs (instead of defined community structures) containing all query keywords [13, 16, 20]. The main novelty of our contextual model is that it resembles k-core and k-truss communities through edge and triangle weighted density w.r.t query context. This enables structure and context cohesiveness simultaneously. The proposed model can advance many applications in a friendly way, such as event scheduling, product recommendation, targeted advertisement, activism and advocacy that leverage the query context.

Structure cohesiveness. Given a query context, inherent contradiction exists with k-core and k-truss models: 1) a larger k value may imply a smaller community that is less informative w.r.t. query context; 2) a smaller k may allow the resultant vertices to fully cover the query context but it may lead to low structure cohesiveness. It is therefore challenging to specify a suitable parameter k. Instead, our proposed contextual community model adopts the parameter-free measure of relative weighted subgraph density. In unweighted case, this density translates to the ratio of the total number of considered edge and triangle units/motifs to the number of their induced vertices in the subgraph. As shown in [24], edge

This work is jointly supported by the ARC Discovery Projects under Grant No. DP160102412, DP160102114, DP170104747 and DP180100212. We would like to thank anonymous reviewers for their helpful comments.

‡ This work is done while the author is affiliated with Swinburne University of Technology.

¶ This work is done while the author is affiliated with The University of Western Australia.

2375-026X/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDE.2019.00017

and triangle densities trade off between subgraph size and cohesiveness. Therefore, our proposed contextual community model considers both edge and triangle units as a unified density measure for discovering a large cohesive contextual community.

Context cohesiveness. Context cohesiveness (or attribute cohesiveness w.r.t. query context) of a community measures how closely its community context (e.g. formed by vertex attributes) matches the query context. However, overemphasising the co-relation between vertex attributes and the query context may cause the search to return a small and loosely connected subgraph. This is against the structural requirement of being a community. To avoid such pessimistic situation, in our model context cohesiveness is relaxed by matching query with the enlarged motif (edge and triangle) attributes instead of the conventional vertex attributes. To calculate this, the context match scores between the query and all the motif attributes are summed up. This measure ensures relaxed but strong internal context cohesiveness.

Contextual community (CC). To search a community with the above structure and context cohesiveness, first, a contextual matching score is assigned to each small motif of a subgraph measuring the context prevalence of a motif. Then, our proposed contextual density of a subgraph is calculated as the ratio of the aggregated score over all motifs to the total number of vertices in the subgraph. Finally, the subgraph with the highest density is returned as the search result. An intuition behind the found community is: its members should be involved in many structurally overlapped edge and triangle motifs which are themselves prevalent w.r.t. query context. In real life, these motifs are analogous to mutual friendships and family circles.

Challenges and contributions. CC is based on the weighted densest subgraph model. The most popular algorithm for finding the maximum weighted degree density subgraph is proposed in [8]. It solves the degree density subgraph problem via the min s-t cut strategy on a carefully designed flow network based on the weighted degree density function. This strategy is adopted to find the maximum triangle density subgraph in [24] that uses the similar idea but proposes a more complicated flow network taking into account triangle density. However, these approaches have two drawbacks: the well known $\mathcal{O}(\log |V(G)|)$ min-cut computations time complexity of the algorithms is valid only for unweighted version of the problem [6]; and their designed flow network can only solve either edge or triangle density problems. CC model considers both weighted edge and triangle simultaneously, in which the above proposed flow networks cannot be applied directly.

To address the above limitations, we first propose an alternative framework for finding CC adapted from [4]. The main idea is, the framework iteratively solves a sub-problem that finds a subgraph with the guaranteed higher contextual density until the optimal CC is derived. We show that the sub-problem can be reduced to finding a min s-t cut in our newly proposed flow network. We highlight that this proposed flow network is very different from [8, 24]. Although our flow network is designed for finding CC, it can also be applied to edge/triangle based density function with minor modification. We also prove that the total number of iterations can be bounded by $\mathcal{O}(|V(G)|)$. In addition, the running time of the framework can be improved to be equivalent to the time taken for solving a single sub-problem. This is achieved with an elegant parametric flow network technique. The main idea is that the state of the successive flow networks can be incrementally updated without solving each sub-problem from sketch. To achieve extra runtime scalability, we also propose a fast $\frac{1}{3}$-approximation algorithm that iteratively and greedily removes vertices affecting the least. The algorithm can run in time $\mathcal{O}(|E(G)|\log(|V(G)|)+|Tri(G)|)$ with simple degree and triangle indices.

The main contributions of the paper are summarized:

- We propose and study a novel CC model and CC search problem. (**Section II**)
- We develop an efficient exact algorithm to solve CC search problem in polynomial time. (**Sections III and IV**)
- We provide an approximate solution with an approximation ratio of $\frac{1}{3}$ that significantly accelerates the exact search algorithms. (**Section V**)
- We conduct extensive experiments on real social network datasets and verify the superior effectiveness and efficiency of CC search methods. (**Section VII**)

II. Problem Definition

A. Preliminary

Attributed graph. An attributed graph is denoted as G=(V,E,A), where $V(G)$, $E(G)$, A denote the set of vertices in G, the set of edges in G, and the set of attributes in terms of *keywords* respectively. Each vertex $v\in V(G)$ is attached with a set of attributes $A(v)\subseteq A$. Given $v\in V(G)$, $deg(v,G)$ denotes the degree of v in G and $N(v,G)$ denotes the neighbours of v in G. A triangle in G is a cycle of length 3. A triangle induced on vertices $u,v,w\in V(G)$ is denoted as $\triangle_{uvw}$ and when these vertices are not specified we omit the subscript. Given a subgraph $H\subseteq G$, $Tri(H)$ denotes the set of triangles in H.

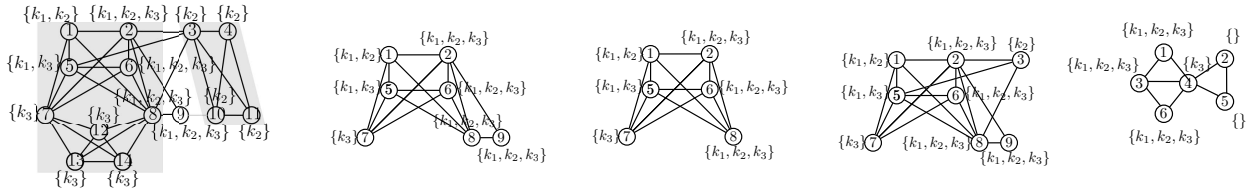
Degree density. Given a subgraph $H\subseteq G$, the degree density of H is defined as $\rho_d(H)=\frac{|E(H)|}{|V(H)|}$. We call it degree density because $|E(H)|=\frac{1}{2}\sum_{v\in V(H)} deg(v,H)$.

Triangle density. Given a subgraph $H\subseteq G$, the triangle density of H is defined as $\rho_\triangle(H)=\frac{|Tri(H)|}{|V(H)|}$.

B. Problem definition

We model a social network as an attributed graph G=(V,E,A), and consider a query context Q as a set of query attributes, where $Q\subseteq A$.

We consider edge and triangle motifs in a social network G as our basic units of a community. To capture both overlaps of the motifs and ensure that the obtained community satisfies a given query context, these basic units are assigned with contextual scores measuring their prevalence to the query context. A subgraph $H\subseteq G$ is considered to have high contextual density if in average every vertex in H is involved in many basic units that are rich in the given query context.



(a) Attributed graph (b) Contextual community (c) Triangle density model (d) Degree density model (e) Sample network

Fig. 1: Example

In the following, we formally define contextual scores of basic motifs, contextual density of a subgraph and contextual community model respectively.

Definition II.1. Contextual score. *Given motifs* $e=(u,v)\in E(H)$, $\triangle=(u,v,w) \in Tri(H)$ *and a set of attributes* Q *as the query context, the contextual score functions for an edge and a triangle denoted by* ES *and* TS *are as follows.*

- *Edge contextual score:* $ES(e,Q) = |Q\cap A(u)|+|Q\cap A(v)|$.
- *Triangle contextual score:* $TS(\triangle,Q)=\sum_{e\in\{(u,v),(u,w),(v,w)\}} ES(e,Q)$.

We highlight the advantages of the proposed contextual score function as follows. Firstly, it awards edge or triangle that is in union covers more query context. For instance, given query $Q=\{k_1,k_2,k_3\}$, triangle $(2,8,9)$ in Figure 1 (a) is superior over triangle $(3,4,11)$ since all query contexts are covered by $(2,8,9)$, which can be differentiated by the defined contextual score. Secondly, it also awards edge or triangle that contains vertices having more query context. For instance, given the same query Q, the defined score also rewards triangle $\{2,8,9\}$ a higher score compared to triangle $\{8,12,14\}$, which makes sense since all vertices in $\{2,8,9\}$ cover all attributes whereas in $\{8,12,14\}$ only vertex 8 covers all attributes. In the remaining for a given query we simply use $ES(e)$ and $TS(\triangle)$ as contextual score functions.

Definition II.2. Contextual density. *Given a set of query attributes, the contextual score functions,* $H\subseteq G$ *with* $Tri(H)$ *containing the set of triangles in* H. *The contextual density of* H, *denoted by* $\rho(H)$ *is defined as:*

$$\rho(H) = \frac{\sum_{\triangle\in Tri(H)} TS(\triangle) + \sum_{e\in E(H)} ES(e)}{|V(H)|}. \quad (1)$$

Problem. CC model and search. *Given a social network graph* G *and a set of query attributes* Q, *return a CC modelled as a subgraph* $H^*\subseteq G$ *satisfying:*

- H^* *is connected.*
- H^* *is* $\arg\max_H\{\rho(H)|H\subseteq G\}$.

Note that $\rho(H)$ can be considered as weighted edge density plus weighted triangle density of H. The weight of the edge and triangle density encourages that the vertices in H are attribute cohesive w.r.t. a query context Q. The edge density encourages large sized H. The triangle density encourages highly cohesive H. As such, maximising $\rho(H)$ encourages communities that attain a proper trade-off over attribute cohesiveness w.r.t. Q, size, and structure cohesiveness.

Example. An attributed graph is shown in Figure 1 (a), which is adopted from [9]. Given $Q=\{k_1,k_2,k_3\}$, applying CC search we can get CC as shown in Figure 1(b). The vertices containing all query contexts are included in the found CC. Compared to CC, if using contextual triangle density only, we get a smaller community, as shown in Figure 1(c), missing vertex 9 that covers all query attributes. On the other hand, if using contextual degree density only, the found community is shown in Figure 1(d), which includes vertex 3 that only covers one query context. Without considering context cohesiveness, the whole graph in Figure 1 (a) excluding edge $(9,10)$ is a connected 3-truss, the grey coloured areas are two connected 4-trusses. We can see that none of them can capture context cohesiveness well.

C. Why contextual community search

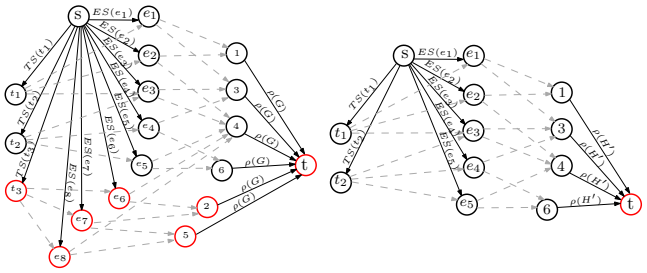
Compared to existing works, our CC model and search are designed to be a general framework for finding attributed communities with several advantages as follows.

- CC employs subgraph density as a parameter-free cohesiveness measure, which avoids specifying k in k-core, k-truss etc., thus it is easy to use.
- CC balances the cohesiveness of the found community by considering both triangles and edges. Triangles reflect denser nature of the community similar to k-truss and edges allow flexibility which somewhat similar to k-core.
- CC search avoids certain bad query effect [11], i.e., the search returns empty set or very loose connected subgraphs because of inappropriate query input. That is because CC simultaneously models structure and contextual cohesiveness.
- CC search indeed has the power to find near ground truth communities as shown in the experimental studies if user-provided query context is close to attributes contained in a ground truth community.

D. Extending the contextual community model

The contextual density of CC model can be extended as the convex combination of triangle and edge contextual densities. As such, we introduce a system configurable parameter $\alpha=[0,1]$ to balance the impact of the triangles and edges in a subgraph. This generalised contextual density function can be formally defined as $\rho_g(H)=\frac{\alpha\sum_{\triangle\in Tri(H)} TS(\triangle)+(1-\alpha)\sum_{e\in E(H)} ES(e)}{|V(H)|}$.

For for simplifying the discussion of proposed search framework, we deliberately focus on the case of $\alpha=0.5$. Also, we would like to highlight that the proposed framework in this



(a) flow network for $\rho(G)$ (b) flow network for $\rho(H')$, H'=$G(\{1,3,4,6\})$

Fig. 2: Running example

TABLE I: Triangle/edge nodes for graph in Figure 1(e)

node	id	contextual score	node	id	contextual score
$\triangle$=(1,3,4)	t_1	$TS(t_1)$=14	$\triangle$=(3,4,6)	t_2	$TS(t_2)$=14
$\triangle$=(2,4,5)	t_3	$TS(t_3)$=2	e=(1,3)	e_1	$ES(e_1)$=6
e=(1,4)	e_2	$ES(e_2)$=4	e=(3,4)	e_3	$ES(e_3)$=4
e=(3,6)	e_4	$ES(e_4)$=6	e=(4,6)	e_5	$ES(e_5)$=4
e=(2,4)	e_6	$ES(e_6)$=1	e=(2,5)	e_7	$ES(e_7)$=0
e=(4,5)	e_8	$ES(e_8)$=1			

paper can be easily generalised to the extended contextual community model with arbitrary value of α.

III. Exact Algorithm

Mathematically the objective function of CC model is to maximize a fractional function. Following this observation, we propose an algorithm solving the CC search problem via an iterative optimization framework. Intuitively, in each iteration the framework is guaranteed to generate a subgraph of higher contextual density untill the global optimal CC is derived. In this section, we focus on explaining how the optimization framework finds the optimum CC and in the next section we show its runtime improvement.

A. Optimization framework

The overall idea of the framework is as follows. We start off by considering the whole graph G as a candidate CC H and then use the stop condition in line 2 of Algorithm 1 (will be explained later) to check its optimality. If not optimum, we generate a better solution H' based on the current H by solving a subproblem $\arg\max_I \{l(\rho(H),I)|I\subseteq G\}$, in which $l(\rho(H),I)$ is defined as:

$$l(\rho(H),I) = \sum_{\triangle\in Tri(I)} TS(\triangle) + \sum_{e\in E(I)} ES(e) - \rho(H)\times(|V(I)|) \quad (2)$$

and check the optimality again. Improved solution gets repeatedly generated as such until the stop condition is met, i.e., a CC H^* is found. These detailed steps are displayed in Algorithm 1.

Example. Let the graph in Figure 1 (e) and Q=$\{k_1,k_2,k_3\}$ be the inputs for Algorithm 1. Initially, we consider H as the whole graph with $\rho(H)$=$\frac{56}{6}$. Suppose we have an $\arg\max_I \{l(\frac{56}{6},I)|I\subseteq G\}$ solver. By solving $\arg\max_I \{l(\frac{56}{6},I)|I\subseteq G\}$, we get vertices $\{1,3,4,6\}$ induced subgraph of G. Clearly, the stop condition is not met. Then we start the iteration, which uses H=$G(\{1,3,4,6\})$ ($\rho(H)$=13), solves $\arg\max_I \{l(13,I)|I\subseteq G\}$ and obtains H'=$G(\{1,3,4,6\})$ again. Then the algorithm terminates since H=H' leads to $l(\rho(H),H')$=0. Algorithm 1 returns H^*=H=$G(\{1,3,4,6\})$ as the optimum result.

Algorithm 1: Iterative optimization algorithm

Data: G, Q
1 $H \leftarrow G$, $H' \leftarrow \arg\max_I \{l(\rho(H),I)|I\subseteq G\}$;
2 **while** $l(\rho(H),H') \neq 0$ **do**
3 $H \leftarrow H'$;
4 $H' \leftarrow \arg\max_I \{l(\rho(H),I)|I\subseteq G\}$;
5 $H^* \leftarrow H$;
6 **return** H^*;

In [6, 12], such framework has been adopted to solve other fractal optimization problems. However, it becomes non-trivial to prove whether the framework together with our carefully designed subproblem solver actually can solve the CC search problem involving contextual scored edges and triangles or not. Further, how fast the problem is solved with the framework is unclear. Specifically, our challenges are: 1) Can lines 2 to 4 in Algorithm 1 ensure H converges to the optimum solution H^* with guarantee? 2) How to solve the subproblem $\arg\max_I \{l(\rho(H),I)|I\subseteq G\}$? And 3) How fast can Algorithm 1 run in the worst case? We answer questions 1) and 2) in the following subsections and 3) in the next section.

B. Algorithm correctness

Algorithm 1 is clearly correct if $\arg\max_I \{l(\rho(H),I)|I\subseteq G\}$ (line 4) always returns H' having a higher contextual density than H, i.e., the current iteration generates a candidate community having the so-far highest contextual density. We formally prove this property with lemmas as follows.

Lemma III.1. *Given H and $H' = \arg\max_I \{l(\rho(H),I)|I\subseteq G\}$ such that $H\neq H'$ then $\rho(H')>\rho(H)$ always holds before CC is generated.*

Proof sketch. By definition, we have $l(\rho(H'),H')$=$\sum_{\triangle\in Tri(H')} TS(\triangle) + \sum_{e\in E(H')} ES(e) - \rho(H')\times(|V(H')|)$ = 0. Then after substracting $\rho(H)\times(|V(H')|)$ from both sides of the equation, we get $\sum_{\triangle\in Tri(H')} TS(\triangle) + \sum_{e\in E(H')} ES(e) - \rho(H)\times(|V(H')|) = \rho(H')\times(|V(H')|) - \rho(H)\times(|V(H')|)$. Therefore if the left side of the equation is greater than 0, then $\rho(H')>\rho(H)$ must hold. For this, we now prove an auxiliary lemma as follows. $\sum_{\triangle\in Tri(H')} TS(\triangle) + \sum_{e\in E(H')} ES(e) - \rho(H)\times(|V(H')|) > 0$, which is equivalent to $l(\rho(H),H')>0$.

Lemma III.2. *Given H and $H' = \arg\max_I \{l(\rho(H),I)|I\subseteq G\}$ such that $H\neq H'$, then $l(\rho(H),H')>0$ always holds before CC is generated.*

Proof sketch. Since $H' \leftarrow \arg\max_I \{l(\rho(H),I)|I\subseteq G\}$, by definition $l(\rho(H),H') > l(\rho(H),H)$ must hold. Since $l(\rho(H),H)$=0 by definition, then $l(\rho(H),H')$ is strictly greater than 0. Hence the correctness of the lemma is clear which ensures the correctness of Lemma III.1. □

Iteration stop condition. Next, we show the stop condition as follows: let $\rho(H^*)$=$\rho(\arg\max_J \{\rho(J)|J\subseteq G\})$, which will be

obtained after certain iterations based on LEMMA III.1, then $l(\rho(H^*),H\leftarrow H')=0$ means H is the result of $\arg\max_J\{\rho(J)|J\subseteq G\}$, which is the optimum CC H^*.

As a conclusion, LEMMAS III.1, III.2 and the stop condition together guarantee the correctness of Algorithm 1.

C. Solving the subproblem

Overall we use maximum flow technique to solve the optimization problem $\arg\max_I\{l(\rho(H),I)|I\subseteq G\}$. We will construct a flow network based on vertices, contextual scored edges, and triangles in G. The $\arg\max_I\{l(\rho(H),I)|I\subseteq G\}$ problem can be reduced to finding the min s-t cut in the constructed flow network. We first revisit the critical definitions and notations of flow network and then discuss the solution in detail.

Directed flow network. The flow network considered in this paper is a directed graph $N=(V,E)$ with a set of nodes V and directed edge set E, having a unique source node s, a unique sink node t, and a non-negative capacity $c(u,v)$ for every directed edge (u,v). Note that we prefer using the term *node* when discussing about flow network, and *vertex* in the context of social network. Following the flow network convention, we extend the capacity function to arbitrary node pairs by defining $c(u,v)=0$, if $(u,v)\notin E(N)$, which implies $f(u,v)=0$ if $(u,v)\notin E(N)$.

Flow. A flow f on N is real-valued function on node pairs satisfying three constraints. First, the capacity constraint, i.e., $f(u,v)\leq c(u,v)$ for every $(u,v) \in V(N) \times V(N)$. Second, the anti-symmetry constraint, i.e., $f(u,v)=-f(v,u)$ for every $(u,v) \in V(N) \times V(N)$. Third, the conservation constraint, i.e., $\sum_{u\in V(N)} f(u,v)=0$ for every $v \in V(N) \setminus \{s,t\}$. The value of the flow f is defined as $\sum_{v\in V(N)} f(v,t)$. A maximum flow is a flow of maximum value.

s-t cut. If S and T are two disjoint node subsets such that $S\cup T=V(N)$ and $S\cap T=\emptyset$, then the capacity or the cut value across the cut (S,T) is $c(S,T)=\sum_{v\in S,u\in T} c(u,v)$. (S,T) is an s-t cut if the source node $s\in S$ and the sink node $t\in T$. A minimum or min s-t cut is a cut with the minimum cut value $c(S,T)$.

Here we show and prove the optimization problem $\arg\max_I\{l(\rho(H),I)|I\subseteq G\}$ can be solved by finding min s-t cut in our carefully constructed flow network.

Flow network construction. Next, we show the detailed construction of the flow network N given a H and the social network G. We firstly create s and t nodes. Then for each triangle, edge and vertex in G we create a triangle node, an edge node and a vertex node. For each triangle node $\triangle$, we create a direct edge from s to the node with capacity of $TS(\triangle)$. Similarly, for each edge node e, an edge is created from s with capacity of $ES(e)$. Then for each triangle node to every edge node involved in the triangle, we create an edge from the triangle node to the edge node with infinite capacity. Similarly, for each edge node to every vertex node involved the edge, we created an edge with infinite capacity from the edge node to the vertex node. At last, for each vertex node, we create an edge from the vertex to t with capacity of $\rho(H)$.

Example. We use an example to show the construction. Let the input graph be the one in Figure 1 (e) and the input query is $Q=\{k_1,k_2,k_3\}$. The triangle and edge nodes are listed in Table I. The flow network for $H=G$ is shown in Figure 2 (a). The flow network for $H=G(\{1,3,4,6\})$ is shown in Figure 2 (b). In them, edges with infinite capacity are drawn as grey dashed arrow lines.

Next we show the equivalence between the problem of solving $\arg\max_I\{l(\rho(H),I)|I\subseteq G\}$ and the problem finding the min s-t cut in N. We achieve this by the following lemmas.

Lemma III.3. ***Valid cut***. *For every s-t cut in N without edges having infinite capacity, nodes in S correspond to a valid subgraph H in G, i.e., (1) if an edge node is in S the involved two vertex nodes are in S and (2) if a triangle node is in S the involved three vertex and edge nodes are in S.*

LEMMA III.3 is obvious from the construction of flow network because of the created edges with infinite capacity.

Lemma III.4. *The min s-t cut for N corresponds to a solution of $\arg\max_I\{l(\rho(H),I)|I\subseteq G\}$.*

Proof sketch. Firstly, the min s-t cut must be a valid cut defined by LEMMA III.3. Then we show the following correspondence. Mathematically, let (S_m,T_m) be the min s-t cut while let (S,T) be a valid s-t cut. Since both of them are valid cut, $c(S_m,T_m) = \sum_{v\in S_m} c(v,t) + \sum_{v\in T_m} c(s,v)$ and $c(S,T)= \sum_{v\in S} c(v,t) + \sum_{v\in T} c(s,v)$. Clearly, the inequality $c(S_m,T_m)\leq c(S,T)$ holds, which is $\sum_{v\in S_m} c(v,t) + \sum_{v\in T_m} c(s,v) \leq \sum_{v\in S} c(v,t) + \sum_{v\in T} c(s,v)$ holds.

Now, by the flow network we constructed, $\sum_{v\in T_m} c(s,v)$ can be expressed as $\sum_{\triangle\in Tri(G)} TS(\triangle) + \sum_{e\in E(G)} ES(e) - (\sum_{\triangle\in T(S)} TS(\triangle) + \sum_{e\in E(S)} ES(e))$, where $T(S)$ denotes triangles represented by triangle nodes in S and $E(S)$ denotes the set of edges represented by edge nodes in S. Since a min s-t cut is valid cut (defined by LEMMA III.3), vertex, edge and triangle nodes in $S\backslash\{s\}$ can form some H' and H' contains triangles and edges represented by $T(S)$ and $E(S)$. Therefore, we can rewrite the expression into $\sum_{\triangle\in Tri(G)} TS(\triangle) + \sum_{e\in E(G)} ES(e) - (\sum_{\triangle\in Tri(H')} TS(\triangle) + \sum_{e\in E(H')} ES(e))$.

Next we show alternatively the expression of $\sum_{v\in S_m} c((v,t))$. Clearly, if (v,t) exists in $E(N)$, v must be vertex node by construction and they must be contained in H'. As a result, $\sum_{v\in S_m} c((v,t))$ can be expressed as $\sum_{v\in V(H')} \rho(H)$, i.e. $\rho(H)\times|V(H')|$. The left part of the inequality now becomes $\sum_{\triangle\in Tri(G)} TS(\triangle) + \sum_{e\in E(G)} ES(e) - (\sum_{\triangle\in Tri(H')} TS(\triangle) + \sum_{e\in E(H')} ES(e)) + \rho(H)\times|V(H')|$.

By the same approach, the right part of the inequality can be expressed as $\sum_{\triangle\in Tri(G)} TS(\triangle) + \sum_{e\in E(G)} ES(e) - (\sum_{\triangle\in Tri(H'')} TS(\triangle) + \sum_{e\in E(H'')} ES(e)) + \rho(H)\times|V(H'')|$, in which H'' is the subgraph consisting of vertex nodes in S and containing all triangles and edges represented by triangle and edges nodes in S. By negating both sides of the above inequality and after simplification, we get $\sum_{\triangle\in Tri(H')} TS(\triangle) + \sum_{e\in E(H')} ES(e) - \rho(H)\times|V(H')| \geq \sum_{\triangle\in Tri(H'')} TS(\triangle) + \sum_{e\in E(H'')} ES(e) - \rho(H)\times|V(H'')|$. The inequality clearly shows the min s-t is equivalent to finding the solution $\arg\max_I\{l(\rho(H),I)|I\subseteq G\}$. □

From min s-t cut to the solution of subproblem. The vertex nodes contained in S of a min s-t cut are used to derive the solution of a subproblem. That is, these nodes induced subgraph of G is the solution.

Example for solving subproblem. In Figure 2(a), the black vertices are in the S_m of the min s-t cut and the flow network is constructed for solving $\arg\max_I\{l(\rho(G),I)|I\subseteq G\}$. Then result of $\arg\max_I\{l(\rho(G),I)|I\subseteq G\}$ is $H'=G(\{1,3,4,6\})$. Similarly, the flow network for $\arg\max_I\{l(\rho(H'),I)|I\subseteq G\}$ is shown in Figure 2(b), result $H=H'$. Algorithm 1 constructs these two flow networks to derive optimum CC.

D. Analyzing the number of iterations

At most $|V(G)|$ iterations. The number of iterations in Algorithm 1 is at most $|V(G)|$, thereby a first runtime complexity of $\mathcal{O}(|V(G)|)\times$ the time complexity of a min s-t cut is implied. We omit the proof details due to the space limit. Note that we will show in the next section how to avoid this runtime factor $|V(G)|$ via an incremental parametric flow framework.

IV. The Incremental Parametric Maximum Flow Exact Algorithm

From the previous discussion we can observe that Algorithm 1 solves a series of strongly correlated maximum flow problems to find the CC. In this section we show that by modifying the proposed flow network, we can apply the faster parametric maximum flow algorithm which instead incrementally updates a flow network. In addition, we also propose heuristic rule making the new algorithm converge to theoretical guaranteed CC with large size.

Intuition. Algorithm 1 successively computes maximum flow problems that are closely related, i.e. during the course of iteration, the structure of flow network does not change much except the capacities which will depend on the newly generated H'. Indeed, in [6], Gallo et.al. have investigated such facts and proposed the so-called parametric maximum flow technique that keeps and incrementally updates the state of the successive flow networks. It turns out that successively solving maximum flow problems is computationally equivalent to only solving one maximum flow, thereby dramatically speed up the whole computation.

However, it is unclear that if this method can be adopted to solve the CC search problem in a much faster way (avoid n rounds of flow computation in our case). In this section we give an affirmative answer to this through slightly modifying the previously constructed flow network. We also prove that the new network indeed correctly finds CC with the parametric maximum flow algorithm. In the following, we first provide preliminaries about parametric maximum flow and then continue with applying this to solve CC search.

A. Preliminaries

The parametric maximum flow algorithm is based on the preflow algorithm. The preflow algorithm computes a maximum flow in a given flow network. Two concepts are essential before presenting the preflow algorithm, that are *preflow* and *valid labelling*.

Algorithm 2: Parametric preflow algorithm

```
Data: G, Q
H←G;
construct a parametric flow network N^R according to H and set λ=ρ(H);
obtain H' from min s-t cut in N^R;
while l(ρ(H),H')≠0 do
    H←H', λ ← ρ(H);
    obtain the largest H' from T of the min s-t cut in N^R ;
H*←H;
return H*;
```

Preflow. A preflow f on N is a real-valued function on node pairs satisfying the capacity and antisymmetry constraints discussed in Section III-C, and the relaxation of the conservation constraint is defined below:

$$\sum_{u\in V(N)} f(u,v)\geq 0 \quad \text{for all } v\in V\setminus\{s\}. \tag{3}$$

Given a preflow, the excess $e(v)$ is defined for v as $\sum_{u\in V(N)} f(u,v)$ if $v\neq s$, or infinity if $v=s$. The value of the preflow is defined as $e(v)$. A node $v\in V(N)\setminus\{s,t\}$ is called active if $e(v)\geq 0$. A preflow is a flow if and only if $e(v)=0$ for all $v\in V(N)\setminus\{s,t\}$. A node pair (u,v) is a residual directed edge for f if $f(u,v)<c(u,v)$, and the difference $c(u,v)-f(u,v)$ is the residual capacity of the directed edge. A pair (u,v) that is not a residual directed edge is saturated.

Valid labelling. A valid labelling d for a preflow f is a function from the nodes to the non-negative integers or infinity, such that $d(t)=0$, $d(s)=|V(N)|$, and $d(u)\leq d(v)+1$ for every residual directed edge (u,v). The residual distance $d_f(u,v)$ from a node v to u is the minimum number of directed edges on a residual path from u to v, or infinity if no such path is available. If d is a valid labelling, $d(v)\leq min\{d_f(v,t),d_f(v,s)+n\}$ for any node v. The intent of such labelling $d(v)$ is to estimate the shortest distance from the vertex v to s or t in a residual network [3]. More specifically, such valid labelling ensures (1) if $d(v)<n$, the distance from v to t is at least $d(v)$ in the residual network and (2) if $d(v)\geq n$ the distance from v to s is at least $d(v)-n$ in the residual network.

Preflow algorithm. The algorithm consists of repeating one of the two operations on an active node u as follows until there is no active node. When the algorithm terminates, the preflow f becomes a maximum flow.

Push operation. Let v be a forward neighbour of u, if $d(u)=d(v)+1$ and $f(u,v)<c(u,v)$, push the flow valued of $min\{e(u),c(u,v)-f(u,v)\}$ from u to v.

Relabelling operation. If for all v in forward neighbours of u, $d(u)\leq d(v)$, relabel $d(u)$ to $min\{d(v)|v\in$forward neighbours of $u\}+1$.

Computation of min s-t cut. After running the preflow algorithm, a minimum cut can be found as follows. For each node $u\in V(N)$, replace $d(u)$ by $min\{d_f(u,s)+|V(N)|,d_f(u,t)\}$. Then, the cut (S,T) defined by $S=\{u|d(u)\geq|V(N)|\}$ is a minimum cut whose sink partition T is of minimum size. If desired, a cut (S',T') of minimum-size S' can be computed as follows. For each $u\in V(N)$ let $d'(v)=min\{d_f(s,u),d_f(t,u)+|V(N)|\}$, and let $S'=\{d'(u)<|V(N)|\}$.

B. Parametric flow framework

Algorithm 2 shows how to find CC using a tailored parametric preflow algorithm. It considers the progressively modified $\rho(H)$ as a parameterised capacity in N^R, where N^R will be discussed later in detail. The overall structure of the algorithm is similar to Algorithm 1, i.e., it continuously generates H with higher contextual density until reaching the stop condition.

Improvements. Compared to Algorithm 1, Algorithm 2 has two non-trivial improvements. Firstly, during each iteration, Algorithm 2 maintains preflow labels via updating the labels computed from the previous iteration. Further, in order to compute H', preflow value and some edge capacities are updated according to H generated in the previous iteration. More details about the update procedure are discussed next after introducing our newly designed parametric flow network N^R. Secondly, Algorithm 2 uses heuristic rule to generate large size H' greedily. The heuristic rule will be discussed in Section IV-D.

C. CC parametric flow network

In this section, we present the parametric flow network N^R used in Algorithm 2 and prove that it makes Algorithm 2 find CC correctly.

Parametric flow network basics. A parametric flow network is a flow network in which edge capacity is a function of λ having following constraints. Firstly, $c_\lambda(s,v)$ is a nondecreasing function of λ for all $v \neq t$. Secondly, $c_\lambda(v,t)$ is a nonincreasing function of λ for all $v \neq s$. Thirdly, $c_\lambda(u,v)$ is constant for all $u \neq s$ and $v \neq t$. The maximum flow or minimum cut in a parametric network is maximum or minimum w.r.t. a particular value of the parameter λ.

Parametric flow network for CC search. The N^R in Algorithm 2 can be generated by the same approach discussed in Section III-C with three modifications: (1) reversing direction of all edges, (2) exchanging s and t, and (3) keeping the same capacities for all edges except for edges directed from s. The capacities for edges directed from s are considered a function of λ, i.e., $c_\lambda(s,v) = \lambda$ and λ will be updated to some newly generated $\rho(H)$ up to the time. The initial value of λ must be the contextual score of the subgraph used to construct N^R. It is not necessary that the subgraph must be the original graph as shown in Algorithm 2; however, to ensure the correctness, the subgraph must contain the optimum CC. The designed N^R clearly satisfies the capacity constraints of parametric flow network.

To ensure the correctness of Algorithm 2, N^R shall satisfy two constraints below and we prove them with corresponding lemmas.

Objective invariant. The min s-t of N^R shall be equivalent to $\arg\max_I \{l(\rho(H),I) | I \subseteq G\}$.

Valid preflow and labelling invariant. After replacing the capacities of all $\{(s,v)\}$ in $E(N^R)$ with newly generated $\rho(H)$ and setting their flow to be $\rho(H)$, the updated N^R shall still maintain a valid preflow and labelling.

Lemma IV.1. *Finding the min s-t of N^R for H is equivalent to solving* $\arg\max_I \{l(\rho(H),I) | I \subseteq G\}$.

The overall proof idea is similar to the proof of LEMMA III.4. The difference is the solution of $\arg\max_I \{l(\rho(H),I) | I \subseteq G\}$ is derived from T of the min s-t cut.

Lemma IV.2. *After replacing capacities to all $\{(s,v)\}$ in $E(N^R)$ with newly generated $\rho(H')$, and setting their flow to be $\rho(H')$, the updated N^R still maintains a preflow and valid labelling.*

Proof sketch. After calculating $\rho(H')$ using push-relabel algorithm, the preflow in the previous N^R becomes a flow. Since push-relabel algorithm maintains valid labelling during the algorithm and after replacing capacities to all $\{(s,v)\}$ in $E(N^R)$ with newly generated $\rho(H')$, we do not modify the labels, therefore, the valid labelling is still maintained after updating the capacity.

On the other hand, after deriving the new H', N^R should have a maximum flow, i.e., for each node v, $f_{n \in V(N^R)} = 0$ shall hold. After setting flows for all $\{(s,v)\}$ in $E(N^R)$ to be the newly generated $\rho(H')$, the flow in N^R becomes a preflow since $\rho(H')$ is greater than previously generated $\rho(H)$. □

The correctness of LEMMAS IV.1 and IV.2 guarantees that the designed N^R makes Algorithm 2 output CC.

Time complexity. The time complexity of Algorithm 2 is $\mathcal{O}(|V(N)|^3)$, which can be expressed as $\mathcal{O}((|E(G)|+|Tri(G)|)^3)$. The proof is the same as [6]. Note that we will propose pruning rules later which significantly reduce the value $|E(G)|+|Tri(G)|$ in real datasets. As such, the practical performance of Algorithm 2 is much faster than the worst case.

D. Finding large CC

Algorithm 2 is primarily designed for finding community with the highest contextual density. Since adding size constraint makes CC search NP-hard, we instead harness with heuristics for reporting a larger connected CC. Our first step is running a depth first search on H' found by Algorithm 2 and outputting the largest connected component in H' as CC. Then for flow network Algorithm 2, we biasedly report larger H' in every iteration. We achieve this by first ranking min-cuts found by the preflow algorithm and then select the min s-t cut partition containing the largest connected components.

Rank min-cuts. As discussed in Section IV-A push-and-relabel algorithm generates two types of min s-t cut when the maximum flow has been computed. The first type maximises the size of S. The later one maximises the size of T. We denote the first type of cut as (S_M,T) while the second type as (S,T_M). We assign them a score as follows. Let X_{S_M} be the vertex nodes in S_M, the score of S_M is defined as the size of the largest connected subgraph of $G(X_{S_M})$, denoted by $cs(S_M)$. Scores are defined for T, S, and T_M via the same approach.

Heuristics. Greedily, for Algorithm 2, when finding a min s-t cut after computing the maximum flow, we always select the cut having $T' = \arg\max_{T'} \{cs(T') | T' \in \{T_M, T\}\}$.

V. Approximation Algorithm

We have demonstrated that via exact algorithm CC can be found in polynomial time. However, despite the solution optimality the runtime bottleneck lies in finding min s-t cut in a large flow network. For further scalability of CC search, we instead look for a greedy based approximation algorithm that trades off solution accuracy for speed. Our proposed algorithm is inspired from the peeling strategy in [14]. In our case, the algorithm removes the vertex affecting the least contextual score iteratively. The main challenge here is to derive guaranteed approximation ratio and runtime. Next, we show this approximate algorithm in Algorithm 3 followed by its approximate ratio and time complexity.

Theorem V.1. *Algorithm 3 achieves a $\frac{1}{3}$-approximation.*

Proof sketch. Let H^* be the global optimal community and $\forall v \in V(H^*)$ let $Tri(v,H^*)$ be the set of triangles containing v in $Tri(H^*)$. Further, let $E(v,H^*)$ be the set of edges containing v in $E(H^*)$. Since $\rho(H^*) \geq \rho(H')$ where $H'=G(V(H^*)\backslash\{v\})$, we can derive the inequality as below:

$$\sum_{\triangle \in Tri(v,H^*)} TS(\triangle) + \sum_{e \in E(v,H^*)} ES(e) \geq \rho(H^*). \quad (4)$$

Now, let H be the subgraph in the iteration when the first vertex $v \in H^*$ is to be removed. Clearly, $H^* \subseteq H$ and for each vertex $u \in V(H)$, by the greediness of Algorithm 3, the following inequality must hold.

$$\begin{aligned} &\sum_{\triangle \in Tri(u,H)} TS(\triangle) + \sum_{e \in E(u,H)} ES(e) \\ \geq &\sum_{\triangle \in Tri(v,H)} TS(\triangle) + \sum_{e \in E(v,H)} ES(e) \\ \geq &\sum_{\triangle \in Tri(v,H^*)} TS(\triangle) + \sum_{e \in E(v,H^*)} ES(e) \end{aligned} \quad (5)$$

Further by inequalities (4) and (5), we can easily conclude that $\sum_{\triangle \in Tri(u,H)} TS(\triangle) + \sum_{e \in E(u,H)} ES(e) \geq \rho(H^*)$.

The total weights of triangles and edges in H can be expressed as

$$= \sum_{u \in V(H)} (\frac{1}{3} \sum_{\triangle \in Tri(u,H)} TS(\triangle) + \frac{1}{2} \sum_{e \in E(u,H)} ES(e)). \quad (6)$$

$$\rho(H) \times |V(H)|$$

Multiplying both sides by 3, we get the inequality as follows.

$$\begin{aligned} &= \sum_{u \in V(H)} (\sum_{\triangle \in Tri(u,H)} TS(\triangle) + \frac{3}{2} \sum_{e \in E(u,H)} ES(e)) \\ &\geq \sum_{u \in V(H)} (\sum_{\triangle \in Tri(u,H)} TS(\triangle) + \sum_{e \in E(u,H)} ES(e)) \\ &\geq \sum_{u \in V(H)} (\rho(H^*)) \end{aligned} \quad (7)$$

$$3 \times \rho(H) \times |V(H)|$$

From (7) immediately $\rho(H) \geq \frac{1}{3}\rho(H^*)$. □

Time complexity. With simple index structures, the runtime of Algorithm 3 can be bounded by $\mathcal{O}$ $(|V(G)|\ \log(|V(G)|) + |E(G)|\ \log(|V(G)|) + |Tri(G)|)$. Intuitively this is because when removing a vertex v from G, we only need to identify edge and triangle changes for the affected vertices in $N(v,G)$. For this, we can just update the contextual scores of the affected vertices while in practice the number of affected edges and triangles is significantly less than $|V(G)|$.

Algorithm 3: Approximate CC

```
Data: G,Q
H←G, Ĥ←G;
while V(H)≠∅ do
    v' ← arg min_{v∈V(H)} {Σ_{△∈T(v,H)} TS(△) + Σ_{e∈E(v,H)} ES(e)};
    H←H(V(H)\{v'});
    if ρ(H)>ρ(Ĥ) then
        Ĥ←H;
return Ĥ ;
```

VI. Search Space Reduction for Exact Algorithms

In this section, we further study optimisations that can improve the efficiency of exact algorithms empirically. We propose two pruning rules preserving the optimality of CC search.

Pruning Rule 1. *Given a vertex $v \in V(G)$, and given a query context Q, v can be pruned if following two conditions are satisfied simultaneously.*

- $A(v) \cap Q = \emptyset$.
- *for each* $u \in N(v,G)$, $A(u) \cap Q = \emptyset$.

The correctness of the **Pruning Rule** 1 is immediate. Given any H, removing vertices confirming the pruning rule will not decrease the numerator value of $\rho(H)$ but will reduce the denominator of $\rho(H)$.

Pruning Rule 2. *Let $\widehat{H}$ be the approximate CC found by Algorithm 3, a vertex $v \in V(G)$ can be pruned if $\sum_{\triangle \in Tri(v,G)} TS(\triangle) + \sum_{e \in E(v,G)} ES(e) < \rho(\widehat{H})$.*

The correctness of **Pruning Rule** 2 can be proved using the inequality in Equation (4), and the pruning rule can be applied iteratively until all vertices in the remaining graph are against the pruning condition.

VII. Experiments

In this section, we conduct comprehensive and extensive experiments to verify the effectiveness and efficiency of the proposed contextual community model and algorithms. Additionally, we also adapt binary search based densest subgraph algorithm used in [8] for CC search as baseline, denoted by *BinCC*. Different from *MonoCC*, *BinCC* progressively determines whether there exists a subgraph with a guessed contextual score and uses binary search to gradually improve the guessed score until the optimum score is reached. This adaption is non-trivial. Due to the space limit we omit the algorithm details. All the algorithms are implemented in Java and run on a MAC with Intel Xeon (3.8 GHz) and 128GB main memory.

A. Experimental setup

Datasets. We use a list of real-life and synthetic datasets in experiments, as shown in Table II. These selected datasets are often applied to evaluate existing methods of addressing the community search, attributed community search, and keyword search problems. *Facebook* is one attributed network dataset

TABLE II: Statistic information of datasets

Dataset	#vertices	#edges	#attributes	# avg. triangles
Facebook	1.9K	8.9K	3064	43
DBLP[1]	977K	3.5M	34213	121
DBpedia	8M	72M	45328	79
DBLP[2]	1.5M	2.4M	13064	32
Amazon	335K	926K	1674	19
DBLP[3]	317K	1M	1584	133
Youtube	1.1M	3M	5327	42
LiveJournal	4M	35M	11104	213
Orkut	3.1M	117M	9926	55
Foursquare	5M	28M	4,970	16

with ground-truth communities. The second line of datasets including *DBLP*[1], and *DBpedia* and *DBLP*[2] are attributed network datasets without the ground-truth communities, which have been used in previous works [5, 13].

We also generate two types of synthetic attributed network datasets, in which attribute are generated synthetically. For the datasets *Amazon*, *DBLP*[3], *Youtube*, *LiveJournal*, and *Orkut*, they have the ground-truth communities, but do not include the attribute information in the vertices. To enrich the attribute information for the network datasets, we first generate a keyword pool, and then randomly select 7 keywords for each community from the keyword pool. After that, we assign each vertex with 0-7 of the keywords related to the community containing the vertex. For the community members, 80% of them must contain at least one of the selected keywords. The above synthetic data generation is similar to that in [11]. For the dataset *Foursquare*, we enrich its attribute information with the approach discussed above for efficiency evaluation only.

Compared models. To verify the performance of our proposed algorithms and community model, we also implemented variants of our model and the other state-of-the-art works. All the compared models (algorithms) have been provided in Table III.

Contextual degree density (ECC). This model is a variant of contextual community model, which uses the contextual scored edges as graph motif to identify the most cohesive community H^* satisfying $H^*=\arg\max_J\{\frac{\sum_{e\in E(J)} ES(e)}{|V(J)|}|J\subseteq G\}$.

Contextual triangle density (TCC). The contextual scored triangles are only used as the graph motif to find the most cohesive community H^* satisfying $H^* = \arg\max_J\{\frac{\sum_{\triangle\in Tri(H)} TS(\triangle)}{|V(J)|}|J\subseteq G\}$.

Attributed truss (ATC in [11]). We also implemented the best algorithm proposed in [11], denoted by *ATC* in this work, to search the attributed truss. Given a set of query vertices, a set of query attributes, two integers k and d, *ATC* is to find the (k,d)-truss satisfying constraints: (1) every edge in the truss has at least (k-2) common neighbours; and (2) the longest shortest path in the (k,d)-truss is no greater than d.

Discrepancy maximisation model [7]. We implemented the exact algorithm in [7] for finding a subgraph that maximises discrepancy, i.e., given a query vertex set Q, finds a subgraph C maximising $\alpha|Q\cap V(C)| - |V(C)\backslash Q|$ (α is set to be 1). To adapt this model to being sensitive to query context, we borrow the method from [7], i.e., given a set of query contexts, take vertex containing all query attributes as the set of query vertices. This method is denoted as *DLAMALL*. To make the comparison more comprehensive, we also consider query vertices as vertices containing at least one of the query contexts as query input and this method is denoted as *DLAMONE*.

Wiener index based model [21]. We implemented an exhaustive search based algorithm for finding a subgraph minimising Wiener index while containing all query vertices. The Wiener index for the subgraph is defined as the sum of the pairwise shortest-path distance in the subgraph. We apply the same approach used in discrepancy maximisation model to make Wiener index based model sensitive to query context resulting in two methods denoted as *MWCALL* and *MWCONE*.

Test queries. For each dataset, we randomly pick up and test 200 keyword queries for each experiment. Each keyword query Q may contain 1-7 terms. The default keyword queries only contain four terms in each of them. We report the average performance of running the 200 keyword queries as the result in the experimental evaluation. To increase the quality of the selected keyword queries, we select the query terms from the attributed vertices appearing in the ground-truth communities if the dataset has the ground-truth information. For the other datasets, we randomly select the query terms and generate the test keyword queries.

Metrics. We use three evaluation metrics to verify the effectiveness of the proposed contextual community model by simulating different query scenarios.

F1 score. We use F1 score metric to evaluate the quality of the resulting communities if the dataset has the ground-truth communities. Given a resulting community H^* and a ground truth community H_g to be targeted, the $F1$ is defined as $F_1\ (H^*,H_g) = \frac{2prec(H^*,H_g)\times recall(H^*,H_g)}{prec(H^*,H_g)+recall(H^*,H_g)}$, where $prec\ (H^*,H_g)$ is the average of $precV\ (H^*,H_g)$ and $precE(H^*,H_g)$, while $recall(H^*,H_g)$ is the average of $recallV\ (H^*,H_g)$ and $recallE\ (H^*,H_g)$ defined as follows. The precision for vertex(edge) is $precV(H^*,H_g) = \frac{|V(H^*)\cap V(H_g)|}{|V(H^*)|}$ ($precE(H^*,H_g) = \frac{|E(H^*)\cap E(H_g)|}{|E(H^*)|}$) . The recall for vertex (edge) is measured by $recallV(H^*, H_g) = \frac{|V(H^*)\cap V(H_g)|}{|V(H_g)|}$ ($recallE(H^*,H_g)=\frac{|E(H^*)\cap E(H_g)|}{|E(H_g)|}$).

Edge density and Jaccard similarity. For the other datasets without the ground truth community information, we measure the average edge density and Jaccard similarity. The edge density measures the cohesiveness of a community H, defined by $ed(H)=\frac{2|E(H)|}{|V(H)|(|V(H)|-1)}$. The Jaccard similarity reflects the attribute cohesiveness of pairwise vertices from the perspective of query context Q, which is defined as $J(u,v)=\frac{|Q\cap A(u)\cap A(v)|}{A(u)\cap A(v)}$.

B. Effectiveness evaluation

1) Dataset with both attributes and ground truth communities: The *Facebook* dataset has 10 ground-truth communities and their attribute information. Figure 3 shows the F1 score of this dataset in the experiment. From the results, we can see that the community found by *CC* method is clearly superior over all the other models. For the ground truth communities f_4 and f_8, *CC* can find the communities with the 90% accuracy that is close to the ground-truth communities. The F1 scores for the communities returned by *ACC* method are sightly lower than that of the communities found by *CC*. The reason is that *ACC* returns approximate *CC* having lower contextual

TABLE III: Implemented algorithm for different community models

Model	Algorithm Notation	Detail
CC	*BinCC*	Implemented binary search based algorithm for searching CC
	MonoCC	Implemented Algorithm 2 for searching CC
	MonoCC+	Implemented Algorithm 2 for searching CC with pruning rules
ACC	*ApxCC*	Implemented Algorithm 3 for searching approximate CC
ECC	*MonoCC*	Implemented Algorithm 2 for searching maximum contextual degree density community
TCC	*MonoCC*	Implemented Algorithm 2 for searching maximum contextual triangle density community
ATC	*ATC*	Implemented LocATC[11] for searching attributed truss
DLAM	*DLAMALL*, *DLAMONE*	Implemented exact local-access model [7]
MWC	*MWCALL*, *MWCONE*	Implemented exact minimum Wiener connector [21] using exhaustive search

density compared to *CC*. Based on the reported F1 scores, *ACC* performs better than *ATC* only except f_6. The major reasons that lead to the lower performance of *ATC* are 1) *ATC* finds the approximate communities based on their model but without theoretical guarantee; and 2) the structural cohesive measurement of *ATC* may discard many edges contained in the ground truth communities but not satisfying the minimum number of common neighbours constraint. *ECC* has the moderate effectiveness in terms of F1 score in this experiment. The reason is that *ECC* tends to find large communities which may result in the much lower precision. Similarly, *TCC* cannot find the communities that is relevant to the ground-truth communities. This is because the model of *TCC* tends to find communities with high precision score, which may lead to the small size of the resulting communities. The small sized communities often have lower score in recall. Hence, the *TCC* method has low F1 scores. *DLAMONE* (*DLAMALL*) can find community closer to ground-truth communities compared with *MWCONE* (*MWCALL*). The reason behind this is as follows. For Wiener index based methods, the number of vertices matching the query attributes is massive since they enforce that the result must contain all matched vertices, so the found community is significantly larger than the size of the ground truth community, which also results in its low precision. In contrast, *DLAMONE* and *DLAMALL* do not enforce the result containing all matched query vertices, so its precision is higher. Compared with the two models (*CC* and *ATC*) that well consider both structure cohesiveness and attribute cohesiveness, *DLAMONE*, *DLAMALL*, *MWCONE* and *MWCALL* cannot find communities closer to the ground-truth communities. This is because in real datasets, the ground-truth communities are cohesive from both structure and attribute perspectives. However, *DLAMONE*, *DLAMALL*, *MWCONE* and *MWCALL* were designed for linking all or part of the query vertices with as less cost (extra vertices) as possible and the attribute-filtering-based adaption to these two models cannot balance the structure cohesiveness and attribute cohesiveness effectively.

2) Dataset with the ground-truth communities only: Figure 4 shows the experimental results when we run the queries on the five datasets, i.e., *Amazon*, and *DBLP*, *Youtube*, *LoveJournal*(*LJ*) and *Orkut*. Our proposed methods can find the communities that are closest to the ground-truth communities. The average F1 score of *CC* is over 90%. Although the average F1 score of *ACC* is about 80%, it still outperforms *ATC* for most of the datasets except for *DBLP*. The main reasons of our methods achieving the superiority are similar to the explanation in Section VII-B1.

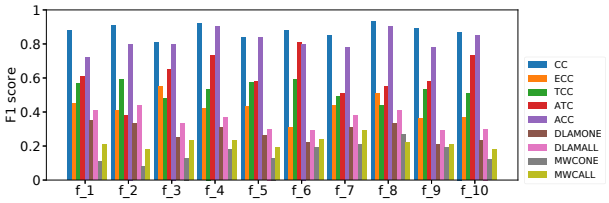


Fig. 3: F1 scores for Facebook

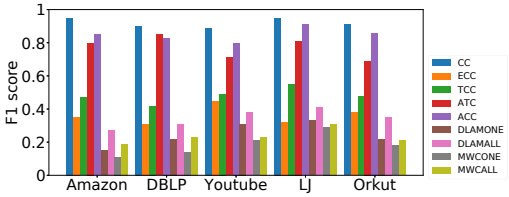
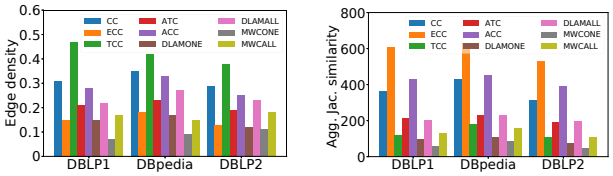
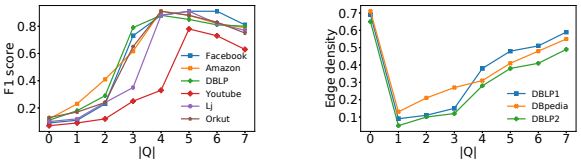


Fig. 4: Effectiveness evaluation



(a) Edge density (b) Aggregate Jaccard similarity

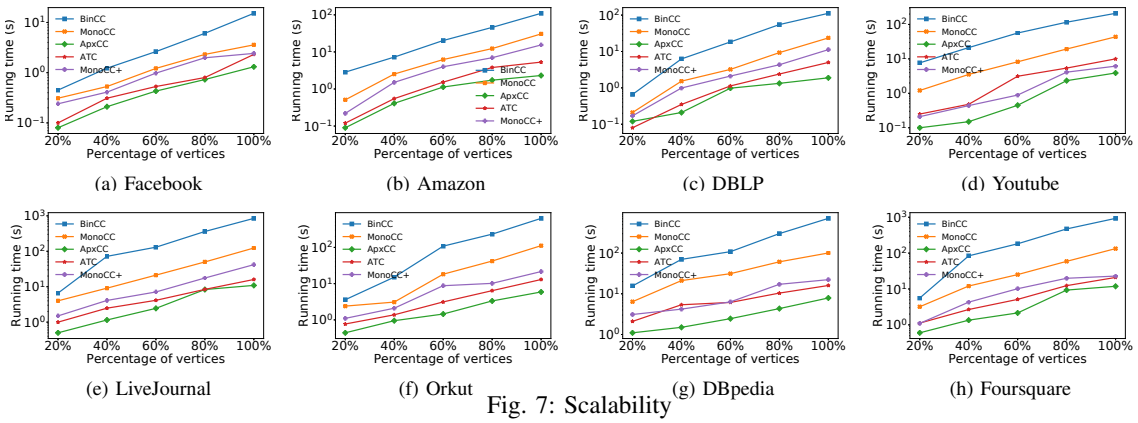
Fig. 5: Attributed networks with no ground-truth



(a) F1 score (b) Edge density

Fig. 6: Sensitivity w.r.t. query context size

3) Datasets with the real attributes: We also evaluate the proposed model using the datasets with real attributes, i.e., *DBLP*[1], and *DBpedia* and *DBLP*[2]. Figure 5 (a) and (b) illustrate the evaluation of the different methods using the metrics of edge density and aggregate Jaccard similarity with regards to different queries. The resulting communities with high edge density and aggregate Jaccard similarity will be treated as the ideal results because such community is more cohesive in structure and context, and its community size is larger. From the results, we can see that *TCC* can find the communities with the highest edge density for all datasets. The edge density can reach 0.45 in average and 0.5 for *DBLP*[1]. But their aggregate Jaccard similarity value is low due to the small size of the communities to be returned. Another observation is that *ECC* can find the communities with the highest aggregate Jaccard similarity value, but the lowest edge density value for all datasets by comparing with



(a) Facebook (b) Amazon (c) DBLP (d) Youtube

(e) LiveJournal (f) Orkut (g) DBpedia (h) Foursquare

Fig. 7: Scalability

all the other methods. It says that the communities of *ECC* may have larger size and higher relevance w.r.t. query but are very sparse from structural perspective. Furthermore, we can see that *CC* and *ACC* have the balanced performance in both high edge density and high aggregate Jaccard similarity. We also find that the communities of *ATC* have less edge density and lower aggregate Jaccard similarity by comparing with *CC* and *ACC*. As shown in Figure 5 (a), *DLAMONE* (*DLAMALL*) can find communities with higher edge density compared with *MWCONE* (*MWCALL*). This is because *DLAMONE* (*DLAMALL*) do not enforce the result to contain all matched query vertices. When comparing with density based models (*CC* and *TCC*) and truss based model (*ATC*), *DLAMONE*, *DLAMALL*, *MWCONE* and *MWCALL* are less effective to find structure dense communities that also satisfy the constraint of query attributes. This is because *DLAMONE*, *DLAMALL*, *MWCONE* and *MWCALL* are based on were mainly designed for linking more query vertices with less non-query vertices, they may not take extra care of the structure cohesiveness. As shown in Figure 5 (b), when comparing with models carefully considering attribute cohesiveness, structure cohesiveness and recall (*CC* and *ATC*), *DLAMONE*, *DLAMALL*, *MWCONE* and *MWCALL* are less effective to find large while highly attribute cohesive communities because their models do not take particular care of both attribute cohesiveness and community member recall simultaneously.

4) Varying Query Sizes: Figure 6 (a) reports F1 score sensitivity w.r.t. the size of query context for *CC* method. For all datasets, F1 score grows as $|Q|$ increases and then decreases after certain threshold. The intuition is that we can find near ground-truth communities if the query context correctly describes the desired communities. The small size of Q implies that the queries cannot precisely describe the ground-truth communities while the large size Q induces noise disturbing the search. The experiment shows that the *CC* method can find the results near to the ground-truth communities in all datasets when the query context includes 4 to 5 attribute keywords.

Figure 6 (b) reports the edge density sensitivity of the *CC* method when we vary the size of query context. For all the datasets, the edge density value increases as $|Q|$ becomes larger. When $|Q|$=0, we consider the unweighted subgraph, which results in the highest edge density. Since the context constrained subgraphs would always be the sub-components of the unweighted subgraph, the context constrained subgraphs become close to the unweighted subgraph when $|Q|$ increases, which aligns with the observed trends.

C. Efficiency evaluation

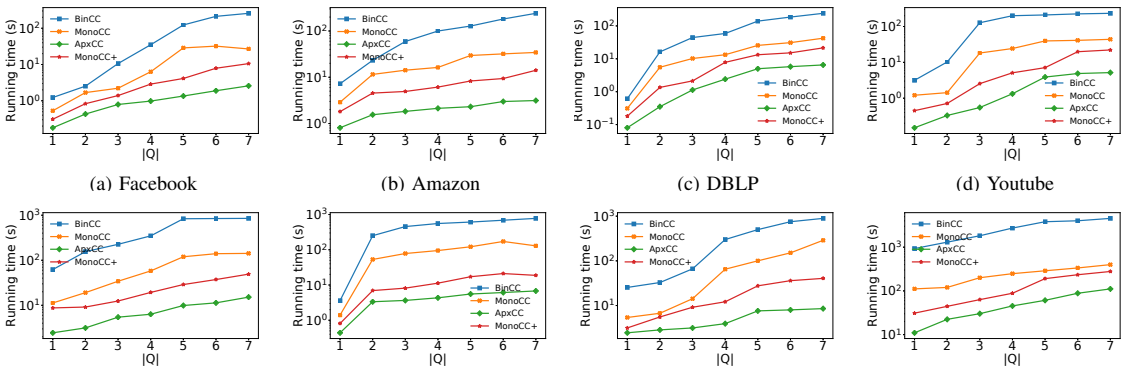
Scalability. Figures 7 (a) to (h) show scalability evaluations for different datasets. For all datasets, our proposed methods *MonoCC+*, *MonoCC* and *ApxCC* perform well as data size increases. Especially for *ApxCC*, it can answer queries within several seconds for all datasets and it is faster than *ATC* for all datasets. Please be noticed that *ATC* is a greedy algorithm with no effectiveness guarantee. *MonoCC* can find exact *CC* within few ten of seconds in most datasets. *MonoCC+* also performs well for all datasets and it only spends 3 to 4 times of time compared with *ApxCC*, which justifies the phenomenal pruning effectiveness of the proposed pruning rules. For all our proposed methods *BinCC*, *MonoCC* and *ApxCC*, the experiment shows they match the discussed time complexity.

Varying $|Q|$. Figures 8 (a) to (h) show the running times as $|Q|$ varies for different datasets. *MonoCC* outperforms *BinCC* 4 to 7 times in average for all datasets, which demonstrates the power of parametric algorithm. *ApxCC* and *MonoCC+* perform much better and the running time grows almost linear as $|Q|$ increases for all datasets. For *MonoCC* and *BinCC*, the running time increases significantly as $|Q|$ becomes greater for all datasets as shown in Figures 8 (a) to (h).

VIII. Related Work

Global community search in attributed graph. Recently Li et al. propose a skyline community model for searching communities in attributed graph [17]. In [27], they find (k,r)-core community such that socially the vertices in (k,r)-core is a k-core and from similarity perspective pairwise vertices similarity is more than a threshold r. CC search is in this category and is different from this model since CC search considers attribute cohesiveness w.r.t. query context and CC search is cohesiveness parameter free.

Node-based community search. In [5, 11, 19], they consider both the structure and keyword cohesiveness when searching



(a) Facebook (b) Amazon (c) DBLP (d) Youtube

(e) LiveJournal (f) Orkut (g) DBpedia (h) Foursquare

Fig. 8: Efficiency evaluation w.r.t. query context size

the community. In [7, 21, 22], they find cohesive subgraphs based on query vertices. CC search is different from their model since CC does not require concrete query vertices.

Community detection in attributed graph. Works such as [18] consider topic LDA model and graph structure to detect attributed communities. Unified distance [29] is also considered for detecting attributed communities. Huang et al. [10] propose a method based on an entropy-based model. Recently, Wu et al. propose an attributed community model [25] based on an attributed refined fitness model. Yang et al. [26] propose a model using probabilistic generative model. Compared to our work these works do not have explicit search intention.

IX. Conclusion

In this paper, we propose a novel parameter-free community model, namely the *contextual community* that only requires a query to provide a set of keywords describing an application/user context. We propose an efficient exact algorithm to solve CC search problem in polynomial time and an approximation algorithm with an approximation ratio of $\frac{1}{3}$. Our empirical studies on real social network datasets demonstrate the superior effectiveness of CC search methods under different query contexts. Extensive performance evaluations also reveal the superb practical efficiency of the proposed CC search algorithms.

References

[1] L. Chen, C. Liu, R. Zhou, J. Li, X. Yang, and B. Wang. Maximum co-located community search in large scale social networks. *PVLDB*, 11(10):1233–1246, June 2018.

[2] J. Cohen. Trusses: Cohesive subgraphs for social network analysis. *National Security Agency Technical Report*, 16, 2008.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[4] W. Dinkelbach. On nonlinear fractional programming. *Management Science*, 13(7):492–498, 1967.

[5] Y. Fang, R. Cheng, S. Luo, and J. Hu. Effective community search for large attributed graphs. *PVLDB*, 9(12):1233–1244, 2016.

[6] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM J. Comput.*, 18(1):30–55, Feb. 1989.

[7] A. Gionis, M. Mathioudakis, and A. Ukkonen. Bump hunting in the dark: Local discrepancy maximization on graphs. In *ICDE*, pages 1155–1166, April 2015.

[8] A. Goldberg. Finding a maximum density subgraph. Technical report, Berkeley, CA, USA, 1984.

[9] 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.

[10] X. Huang, H. Cheng, and J. X. Yu. Dense community detection in multi-valued attributed networks. *Inf. Sci.*, 314(C):77–99, Sept. 2015.

[11] X. Huang and L. V. S. Lakshmanan. Attribute-driven community search. *PVLDB*, 10(9):949–960, 2017.

[12] P. Jean-Claude and Q. Maurice. A network flow solution to some nonlinear 0-1 programming problems, with applications to graph theory. *Networks*, 12(2):141–159, 1982.

[13] M. Kargar and A. An. Keyword search in graphs: Finding r-cliques. *PVLDB*, 4(10):681–692, 2011.

[14] S. Khuller and B. Saha. On finding dense subgraphs. ICALP, pages 597–608, Berlin, Heidelberg, 2009. Springer-Verlag.

[15] T. La Fond and J. Neville. Randomization tests for distinguishing social influence and homophily effects. In *WWW*, pages 601–610, 2010.

[16] J. Li, C. Liu, and M. S. Islam. Keyword-based correlated network computation over large social media. In *ICDE*, pages 268–279, 2014.

[17] R.-H. Li, L. Qin, F. Ye, J. X. Yu, X. Xiao, N. Xiao, and Z. Zheng. Skyline community search in multi-valued networks. SIGMOD, pages 457–472, New York, NY, USA, 2018. ACM.

[18] R. M. Nallapati, A. Ahmed, E. P. Xing, and W. W. Cohen. Joint latent topic models for text and citations. In *SIGKDD*, pages 542–550, New York, NY, USA, 2008. ACM.

[19] B. Perozzi, L. Akoglu, P. Iglesias Sánchez, and E. Müller. Focused clustering and outlier detection in large attributed graphs. In *SIGKDD*, pages 1346–1355. ACM, 2014.

[20] L. Qin, J. Yu, L. Chang, and Y. Tao. Querying communities in relational databases. In *ICDE*, pages 724–735, March 2009.

[21] N. Ruchansky, F. Bonchi, D. García-Soriano, F. Gullo, and N. Kourtellis. The minimum wiener connector problem. In *SIGMOD*, pages 1587–1602, New York, NY, USA, 2015. ACM.

[22] N. Ruchansky, F. Bonchi, D. Garcia-Soriano, F. Gullo, and N. Kourtellis. To be connected, or not to be connected: That is the minimum inefficiency subgraph problem. In *CIKM*, pages 879–888. ACM, 2017.

[23] S. B. Seidman. Network structure and minimum degree. *Social Networks*, 5(3):269 – 287, 1983.

[24] C. Tsourakakis. The k-clique densest subgraph problem. In *WWW*, pages 1122–1132, 2015.

[25] P. Wu and L. Pan. Mining application-aware community organization with expanded feature subspaces from concerned attributes in social networks. *Knowledge-Based Systems*, 139:1 – 12, 2018.

[26] J. Yang, J. McAuley, and J. Leskovec. Community detection in networks with node attributes. In *ICDM*, pages 1151–1156. IEEE, 2013.

[27] F. Zhang, Y. Zhang, L. Qin, W. Zhang, and X. Lin. When engagement meets similarity: efficient (k, r)-core computation on social networks. *PVLDB*, 10(10):998–1009, 2017.

[28] R. Zhou, C. Liu, J. X. Yu, W. Liang, B. Chen, and J. Li. Finding maximal k-edge-connected subgraphs from a large graph. In *EDBT*, pages 480–491, 2012.

[29] Y. Zhou, H. Cheng, and J. X. Yu. Graph clustering based on structural/attribute similarities. *PVLDB*, 2(1):718–729, 2009.