Discovering Nested Communities

Nikolaj Tatti and Aristides Gionis

Helsinki Institute for Information Technology
Department of Information and Computer Science
Aalto University
{nikolaj.tatti,aristides.gionis}@aalto.fi

Abstract. Finding communities in graphs is one of the most well-studied problems in data mining and social-network analysis. In many real applications, the underlying graph does not have a clear community structure. In those cases, selecting a single community turns out to be a fairly ill-posed problem, as the optimization criterion has to make a difficult choice between selecting a tight but small community or a more inclusive but sparser community.

In order to avoid the problem of selecting only a single community we propose discovering a sequence of nested communities. More formally, given a graph and a starting set, our goal is to discover a sequence of communities all containing the starting set, and each community forming a denser subgraph than the next. Discovering an optimal sequence of communities is a complex optimization problem, and hence we divide it into two subproblems: 1) discover the optimal sequence for a fixed order of graph vertices, a subproblem that we can solve efficiently, and 2) find a good order. We employ a simple heuristic for discovering an order and we provide empirical and theoretical evidence that our order is good.

Keywords: community discovery, monotonic segmentation, graph mining, nested communities.

1 Introduction

Discovering communities, tightly connected subgraphs, is one of the most well-studied problems in the field of graph mining. Given some optimization criterion, discovering a community is a computationally challenging task, typically **NP**-hard. Additionally, as pointed out by Leskovec et al. [17], in many real applications the underlying graph does not have a clear community structure. Such cases make the community-finding problem inherently ill-posed, as the optimization criterion has to make a difficult, and eventually arbitrary, choice between selecting a tight but small community or a more inclusive but more sparse community. Moreover, the existence of a universal criterion for making such a choice is unlikely as the balance between the size and the density of the desired community will depend on the underlying application.

In order to avoid the problem of selecting only a single community, we propose a problem of discovering a *sequence of nested communities*. More formally, given

H. Blockeel et al. (Eds.): ECML PKDD 2013, Part II, LNAI 8189, pp. 32–47, 2013.

a graph G and a set of source vertices S, our goal is to discover a sequence of k communities around S, such that each community is a subset of the next one. The first community will consist only of S while the last community will contain the whole graph. Inner communities should be tighter than the outer communities. We express this requirement by computing the density of each community and require that the next community should have a lower density than the current community. In addition, we require that each community should be as uniform as possible. We measure uniformity by computing the variance of weights of the edges and requiring it to be small.

Discovering a sequence of communities by optimizing the uniformity criterion is a challenging problem. We will show that several optimization problems related to the optimal solution are **NP**-hard. Hence, we split the problem into two subproblems. We can view a community sequence as a bucket order on the vertices, each bucket consisting of vertices contained in the community and not contained in the previous community. Our first subproblem is to discover a total order on the vertices respecting the optimal bucket order. The second subproblem is to discover the optimal sequence of communities, given an order on the graph vertices. Fortunately, this subproblem can be formulated as a standard sequence-segmentation problem, and thus, it can be solved in polynomial time. In particular, we can solve this problem optimally in quadratic time or we can find an approximate solution in nearly-linear time. Discovering the order is more difficult as this is a complex combinatorial problem. We propose a simple ordering technique used for discovering dense subgraphs: pick iteratively a vertex with the lowest degree, and remove it from the graph. We provide theoretical evidence implying that this is a good order and we also show experimentally that this order outperforms several baselines.

The rest of the paper is organized as follows. We introduce preliminary notation in Section 2 and formalize our optimization problem in Section 3. In section 4 we develop our discovery algorithm and point out theoretical properties of our approach. Section 5 is devoted to related work and Section 6 is devoted to experimental evaluation. We conclude our paper with a short conclusion in Section 7.

2 Preliminaries

We consider a weighted undirected graph $G = (V, E, w)$ over a set of vertices V and edges $E \subseteq \binom{V}{2}$. We use the notation $\binom{V}{2}$ to denote the set of unordered pairs of distinct vertices from V. The function $w : E \to \mathbb{R}$ assigns a weight $w(e)$ to each edge $e \in E$. Also, given a subset of vertices $V' \subseteq V$ we denote by $E(V')$ the set of edges in the *induced* subgraph of G defined by V'.

The definitions and algorithms in this paper rely on a notion of *edge density*, which is defined not only over subsets of vertices, but also over arbitrary *pairs* of subsets of vertices. Even though it is conceptually simple, our edge-density definition requires slightly complex notation for determining the set of potential edges to be used as a denominator in the density ratio. To simplify our presentation we use the notation described below.

Given the graph $G = (V, E, w)$, we consider its *completed* representation $G_0 = (V, E_0, w_0)$, where $E_0 = \binom{V}{2}$, and where w_0 is an extension of w, so that $w_0(e) = w(e)$ if $e \in E$, and $w_0(e) = 0$ if $e \notin E$. In other words, G_0 can be seen as a complete graph, where all non-edges of G become zero-weight edges in G_0. We note again that we use the completed graph representation only to simplify our notation; in our implementation there is no need to store the zero-weight edges.

Now consider the completed representation $G_0 = (V, E_0, w_0)$ of a graph G, and let $F \subseteq E_0$ be a non-empty subset of edges. We define the *weight* and *density* of F as

$$w(F) = \sum_{e \in F} w(e) \quad \text{and} \quad d(F) = \frac{w(F)}{|F|}.$$

Consider now two subsets of vertices $S, T \subseteq V$. We define the set of *cross edges* from S to T as $c(S, T) = \{(x, y) \in E \mid x \in S, y \in T\}$. It is important to note that we do not impose any constraint on the sets S and T; they may overlap in an arbitrary way. For instance, if the sets S and T are disjoint the edges in $c(S, T)$ are the *cut* edges from S to T, while if $S \subseteq T$ the edge set $c(S, T)$ contains, among others, all the edges within S.

Finally, we write $w(S, T)$ as a shorthand of $w(c(S, T))$ and we write $d(S, T)$ as a shorthand of $d(c(S, T))$.

3 Nested Communities

As we discussed in the introduction, our goal is to find the optimal sequence of nested communities, with respect to a set of source vertices of the input graph. We denote this set of source vertices by S. For conceptual simplicity, one may think of S as a singleton set, that is, identifying the sequence of nested communities for a single vertex. However, all our problem definitions, algorithms, and proofs, hold for the general case of S being any subset of V.

Our objective is to find k nested communities, where the parameter k is part of the problem input. Given a set of source vertices S, we represent a sequence of nested communities with respect to S, by the sequence of vertex sets $S = V_0 \subseteq V_1 \subseteq \cdots \subseteq V_k = V$.

Intuitively, the inner sets of the nested-community sequence are expected to be more strongly related to the source set S. This type of relatedness is expressed by the notion of density. So, V_1 is the densest community that contains S, V_2 is the second densest community, and in general, we require that the density of V_i should decrease as i increases.

Considering the requirement of monotonically decreasing density in isolation is not sufficient to determine in a well-defined manner a desirable sequence of nested communities. Indeed, given a graph G, a set of source vertices S, and integer k, there is a potentially exponential number of ways to partition the set of vertices of the graph into a sequence of nested communities $V_0, \ldots, V_k$.

The main question we are facing is to decide where exactly to draw the boundary between each pair of communities V_i and V_{i+1}. To answer this question, we

follow an approach inspired by *segmentation problems*. In particular, our approach is as follows: consider the set of vertices $D_{i+1} = V_{i+1} \setminus V_i$ that need to be added to the community V_i in order to form community V_{i+1}. Consider also the set of edges $E_{i+1} = E(V_{i+1}) \setminus E(V_i)$, defined as the additional edges brought in by extending the community V_i to the community V_{i+1}. We can then define the density of the set of edges E_{i+1}. To capture the intuition that the set D_{i+1} should form a coherent extension to V_i we require that the density of E_{i+1} is as *uniform* as possible.

The notion of uniformity for a set of edges, among many ways, can be expressed as a sum of square of difference of the weight of each edge from the average weight of the set. We thus have the following definition.

Definition 1. *Given a set of edges $F \subseteq E$, we define the* density-uniformity score *as*

$$q(F) = \sum_{e \in F} \left(w(e) - d(F)\right)^2 .$$

Our goal is then to find a sequence of nested communities so that the successive segments of added edges are as uniform as possible with respect to their density. Formulating this objective as an optimization problem not only gives meaningful semantics to the nested community detection problem, but it also makes the problem well-defined. Motivated by the discussion above, our main problem definition is given below.

Problem 1. Given a weighted input graph $G = (V, E, w)$, a set of source vertices $S \subset V$, and an integer k, find the sequence of nested communities $\mathcal{V} = \{S = V_0 \subseteq V_1 \subseteq \cdots \subseteq V_k = V\}$ that minimizes the density-uniformity score

$$q(\mathcal{V}) = \sum_{i=1}^{k} q(E(V_i) \setminus E(V_{i-1})) ,$$

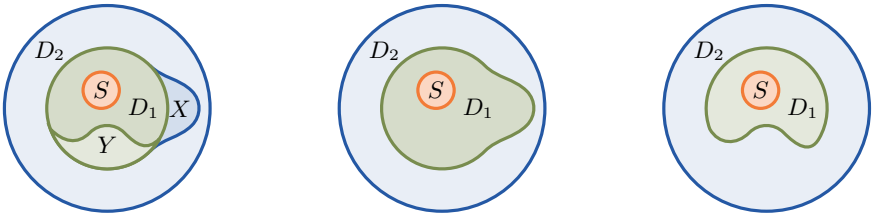
subject to the constraint $d(V_i) < d(V_{i-1})$ for $i = 2, \ldots, k$.

4 An Algorithm for Discovering Nested Communities

In this section we present our algorithm for discovering nested communities. We begin by demonstrating a necessary condition for the optimal solution based on dense subgraphs. Discovering such subgraphs turns out to be computationally intractable. We then split the original problem into two subproblems: discovering community sequence for a fixed order of vertices, a problem which we can solve efficiently, and discovering such an order. We provide a simple heuristic for discovering an order, and provide theoretical evidence that this order is good.

4.1 Nested Communities and Dense Subgraphs

We start our discussion by demonstrating a connection of the problem of finding the optimal sequence of nested communities, i.e., solving Problem 1, with problems related to finding dense subgraphs of a given graph.



(a) Original community (b) Adding X (c) Removing Y

Fig. 1. Communities related to Proposition 1. If $d(X, X \cup D_1) > d(Y, D_1)$, then either adding X to D_1 or removing Y from D_1 will yield a better score.

To establish this connection, consider a triple of communities $V_{i-1} \subseteq V_i \subseteq V_{i+1}$ in an *optimal solution* to Problem 1. Consider the two corresponding segments $D_{i+1} = V_{i+1} \setminus V_i$ and $D_i = V_i \setminus V_{i-1}$. Consider also any two subsets of those segments, $X \subseteq D_{i+1}$ and $Y \subseteq D_i$, that is, X is a subset of the outer segment, while Y is a subset of the inner segment, see Figure 1(a) for a visualization. As we will show shortly, adding the outer subset X in the community V_i leads to a situation where the density of the subset X with respect to the overall community V_i is no better than the density of the subset Y with respect to the community V_i. Otherwise, either adding X to V_i (see Figure 1(b)) or removing Y from V_i (see Figure 1(c)) lead to a better solution. This follows from the fact that we require that the densities of the nested communities in any feasible solution of Problem 1 decrease monotonically.

Before proceeding to discussing the implications of this observation, we first give a formal statement and its proof.

Proposition 1. *Consider a graph $G = (V, E, w)$, a set of source vertices $S \subseteq V$, and an integer k. Let $\mathcal{V} = (S = V_0 \subseteq V_1 \subseteq \cdots \subseteq V_k = V)$ be the optimal sequence of nested communities, that is, a solution to Problem 1. Fix i such that $1 \leq i \leq k-1$ and let $X \subseteq V_{i+1} \setminus V_i$ and $Y \subseteq V_i \setminus V_{i-1}$. Then*

$$d(X, X \cup V_i) \leq d(Y, V_i)\,.$$

For the proof of the proposition we require the following lemma, which states that the mean square error of a set of numbers from a single point, increases with the distance of that point from the mean of the numbers. The lemma can be derived by simple algebraic manipulations, and its proof is omitted.

Lemma 1. *Let $w_1, \ldots, w_N$ and $x_1, \ldots, x_N$ be two sets of real numbers. Let $W = \sum_{i=1}^{N} w_i$ and $\mu = \frac{1}{W} \sum_{i=1}^{N} w_i x_i$. For any real number d it is*

$$\sum_{i=1}^{N} w_i (x_i - d)^2 = \sum_{i=1}^{N} w_i (x_i - \mu)^2 + W(d - \mu)^2.$$

We are now ready to prove the proposition.

Proof (Proposition 1). Let $C_1 = E(V_{i+1}) \setminus E(V_i)$ and $C_2 = E(V_i) \setminus E(V_{i-1})$. Let us break C_1 into two parts, $D_{11} = c(X, X \cup V_i)$ and $D_{12} = C_1 \setminus D_{11}$. Similarly, let us break C_2 into two parts, $D_{21} = c(Y, V_i)$ and $D_{22} = C_2 \setminus D_{21}$. Define the centroids $\mu_{ij} = d(D_{ij})$ and $\lambda_i = d(C_i)$. Lemma 1 now implies that

$$\begin{aligned} s = q(C_1) + q(C_2) &= \text{const} + |D_{11}|(\mu_{11} - \lambda_1)^2 + |D_{21}|(\mu_{21} - \lambda_2)^2, \\ s_1 = q(C_1 \cup D_{21}) + q(D_{22}) &= \text{const} + |D_{11}|(\mu_{11} - \lambda_1)^2 + |D_{21}|(\mu_{21} - \lambda_1)^2, \\ s_2 = q(D_{12}) + q(C_1 \cup D_{11}) &= \text{const} + |D_{11}|(\mu_{11} - \lambda_2)^2 + |D_{21}|(\mu_{21} - \lambda_2)^2, \end{aligned}$$

where const is equal to

$$\sum_{i=1}^{2} q(D_{i1}) + q(D_{i2}) + |D_{i2}|(\mu_{i2} - \lambda_i)^2 \quad .$$

Since $\mathcal{V}$ is optimal we must have $s \leq s_1$ and $s \leq s_2$. Otherwise, we can obtain a better segmentation by attaching X to V_i or deleting Y from V_i. This implies that $|\mu_{21} - \lambda_2| \leq |\mu_{21} - \lambda_1|$ and $|\mu_{11} - \lambda_1| \leq |\mu_{11} - \lambda_2|$. Since $\lambda_2 \geq \lambda_1$, this implies that $\mu_{21} \geq (\lambda_1 + \lambda_2)/2$ and $\mu_{11} \leq (\lambda_1 + \lambda_2)/2$, which implies $\mu_{11} \leq \mu_{21}$. This completes the proof. □

Proposition 1 implies that in an optimal solution the graph vertices can be *ordered* in such a way so that subgraph density, as specified by the proposition, decreases along this order. This observation motivates the following *greedy* algorithm for solving the problem of discovering nested communities:

Algorithm Outline: Greedy–add–densest–subgraph

1. Start with S, the set of source vertices.
2. Given the current set S, find a subset of vertices T that maximize $d(T, S \cup T)$.
3. Set $S \leftarrow S \cup T$, and repeat the previous step until the set S includes all the vertices of the graph.
4. Consider the vertices in the order discovered by the previous process. Find the optimal sequence of k nested communities that respects this order.

One potential problem with the above greedy approach is that the subroutine that is called iteratively in step 2, is an **NP**-hard problem. This is formalized below as problem DenseSuperset.

*Problem 2 (*DenseSuperset*).* Given a weighted graph $G = (V, E, w)$ and a subset of vertices $S \subseteq V$, find a subset of vertices T maximizing $d(T, S \cup T)$.

Proposition 2. *The* DenseSuperset *problem is **NP**-hard.*

Proof (Sketch). Due to space constraints we will only sketch the proof. The complete proof is available in Appendix.[1] We will reduce Clique to DenseSuperset. Given a graph G, we add a vertex s and connect it to each vertex with a weight of $\alpha = 1 - \frac{1}{2|V|^2}$. Let $k < n < m$. It follows that

[1] For the appendix, see `http://users.ics.aalto.fi/~ntatti/`

$$\frac{\binom{n}{2}+\alpha n}{\binom{n}{2}+n} > \frac{\binom{k}{2}+\alpha k}{\binom{k}{2}+k} \quad \text{and} \quad \frac{\binom{n}{2}+\alpha n}{\binom{n}{2}+n} > \frac{\binom{m}{2}+\alpha m-1}{\binom{k}{2}+m} \quad .$$

The left-hand side term in the first equation is the density of n-clique while the the right-hand side term bounds the density of a graph with k vertices. The right-hand side term in the second equation upper bounds the density of a non-clique with m vertices. Consequently, the largest clique, say X, in G will also have the largest density $d(X, X \cup s)$, which is a sufficient to prove the result. □

Similarly, one can think of solving the problem by working on the opposite direction, that is, start with the whole vertex set V and "peel off" the set V by removing the sparsest subgraph, until left with the set of source vertices S. The corresponding algorithm will be the following.

Algorithm Outline: Greedy–remove–sparsest–subgraph

1. Start with V, the vertex set of G.
2. Given a current set V, find a subset of vertices T that does not include the source vertex set S and minimizes the density $d(T, V)$.
3. Set $V \leftarrow V \setminus T$, and repeat the previous step until left only with the set of source vertices S.
4. Consider the vertices in the order removed by the previous process. Find the optimal sequence of k nested communities that respects this order.

Not surprisingly, the problem of finding the sparsest subgraph, which corresponds to step 2 of the above process is **NP**-hard.

*Problem 3 (*SparseNbhd*).* Given a weighted graph $G = (V, E, w)$ find a set of vertices T minimizing $d(T, V)$.

Proposition 3. *The* SparseNbhd *problem is* ***NP****-complete.*

Proof (Sketch). Due to space constraints we will only sketch the proof. The complete proof is available in Appendix. We will reduce Clique to SparseNbhd. Assume that we are given a graph G with l nodes. We extend the graph by adding two vertices s and t with an edge of such high weight that neither s or t will appear in the optimal solution. We then add an edge from s to each vertex v in G with a weight of $p - \deg(v)$, where $p = (l+1) - \frac{1}{2}(l+1)^{-2}$. This will make the weighted degree of all vertices in G equal so a dense subgraph X will have a low density $d(X, X \cup \{s, t\})$. Let $k < n < m$. Then a straightforward calculation reveals that

$$\frac{pn-\binom{n}{2}}{(l+1)n-\binom{n}{2}} < \frac{pk-\binom{k}{2}}{(l+1)k-\binom{k}{2}} \quad \text{and} \quad \frac{pn-\binom{n}{2}}{(l+1)n-\binom{n}{2}} < \frac{pm-\binom{m}{2}+1}{(l+1)m-\binom{m}{2}} \quad .$$

The left-hand side term in the first equation is the density of n-clique while the right-hand side term bounds the density of a graph with k vertices. The right-hand side term in the second equation lower bounds the density of a non-clique with m vertices. Consequently, the largest clique, say X, in G will also have the lowest density $d(X, X \cup \{s, t\})$, which is a sufficient to prove the result. □

4.2 Algorithm for Discovering Nested Communities

Armed with intuition from the previous section, we now proceed to discuss the proposed algorithm. The underlying principle of both of the greedy algorithms described above is to consider the vertices of the graph in a specific order and then find a sequence of nested communities that respects this order. In one case, the order of graph vertices is obtained by starting from S and iteratively adding the densest subgraph, while in the other case, the order is obtained by starting from the full vertex set V and iteratively removing the sparsest subgraph.

Our algorithm is an instantiation of this general principle. We specify in detail (i) how to obtain an order of the graph vertices, and (ii) how to find a sequence of nested communities that respects a given order.

We start our discussion from the second task, i.e., finding the sequence of nested communities given an order. As it turns out, this problem is an instance of sequence segmentation problems. We define this problem below, which is a refinement of Problem 1.

Problem 4 (Sequence of nested communities from a given order). Given a graph $G = (V, E, w)$ *with ordered vertices*, a set of source vertices $S = \{v_1, \ldots, v_s\} \subset V$, and an integer k, find a monotonically increasing sequence of $k + 1$ integers $b = (b_0 = s, \ldots, b_k = |V|)$ such that

$$\mathcal{V} = (S = V_0 \subseteq V_1 \subseteq \cdots \subseteq V_k = V), \quad \text{where} \quad V_k = \{v_1, \ldots, v_{b_i}\},$$

minimizes the density-uniformity score $q(\mathcal{V})$ and satisfies the monotonicity constraint $d(V_i) < d(V_{i-1})$ for $i = 1, \ldots, k$.

It is quite easy to see that Problem 4 can be cast as a segmentation problem. Typical segmentation problems can be solved optimally using dynamic programming, as shown by Bellman [3]. The most interesting aspect of Problem 4, seen as segmentation problem, is the monotonicity constraint $d(V_i) < d(V_{i-1})$, for $i = 1, \ldots, k$. That is, not only we ask to segment the ordered sequence of vertices so that we minimize the density variance on the segments, but we also require that the density scores of each segment decrease monotonically. The situation can be abstracted to the monotonic segmentation problem stated below.

Problem 5 (Monotonic segmentation). Let $a_1, \ldots, a_n$ and $x_1, \ldots, x_n$ be two sequences of real numbers. Given an integer k, find $k + 1$ indices $b_0 = 1, \ldots, b_k = n + 1$ minimizing

$$\sum_{j=1}^{n} \sum_{i=b_{j-1}}^{b_j - 1} a_i (x_i - \mu_j)^2,$$

where μ_j is the weighted centroid of j-th segment such that $\mu_j < \mu_{j-1}$.

In order to express Problem 4 with Problem 5, consider a group of edges, $P_i = c(v_i, \{v_1, \ldots, v_{i-1}\})$ for each vertex $v_i \in V \setminus S$. If we set $a_i = \left|P_{i+|S|}\right|$ and $x_i = d\left(P_{i+|S|}\right)$, we can apply Lemma 1 and show that the score of community

sequence is equal to the variance minimized by Problem 5, plus a constant. In fact, this constant is the sum of the variances within each P_i.

Similarly to the unconstrained segmentation problem, the monotonic segmentation problem can be solved *optimally*. The idea is to use as preprocessing step the classic "pool of adjacent violators" algorithm (PAV) [2], which merges points until there are no monotonicity violations, and then apply the classic dynamic-programming algorithm on the resulting sequence of merged points. This algorithm runs in $O(|V|)$ time. By definition the merged points do not contain any monotonicity violations, and thus, the resulting segmentation respects the monotonicity constraint, as well. As shown by Haiminen et al. [14], this two-phase algorithm gives the optimal k segmentation under the monotonicity constraints. As a result of the optimality of the monotonic segmentation problem, Problem 4 can be solved optimally.

We next proceed to discuss the first component of the algorithm, namely, how to obtain an order of the graph vertices. Recall that, according to the principles discussed in the previous section, we can either start from S and iteratively add dense subgraphs, or start from V and remove sparse subgraphs. We follow the latter approach. In order to overcome the **NP**-hard problem of finding the sparsest subgraph and in order to obtain a total order, we use the heuristic of iteratively removing the sparsest subgraph of size one, namely, a single vertex. The sparsest one-vertex subgraph is simply the vertex with the smallest weighted degree. Thus, overall, we obtain the simple algorithm SortVertices, whose pseudocode is given as Algorithm 1.

As an interesting side remark, we note that the algorithm SortVertices is encountered in the context of finding subgraphs with the highest average degree. In particular, it is known that the densest subgraph obtained by the algorithm during the process of iteratively removing the smallest-degree vertex is a factor-2 approximation to the optimally densest subgraph in the graph [4].

The natural question to ask is how good is the order produced by algorithm SortVertices? As we will demonstrate shortly, it turns out that the order is quite good. First, we note that the optimal solution obtained for Problem 4, satisfies an analogous structural property, with respect to subgraph densities, as the optimal solution for Problem 1, We omit the proof of the following proposition as it is similar to the one of Proposition 1.

Proposition 4. *Consider a graph $G = (V, E, w)$ with ordered vertices, a set of source vertices $S \subset V$, and an integer k. Let $\mathcal{V} = (S = V_0 \subseteq V_1 \subseteq \cdots \subseteq V_k = V)$ be the optimal sequence of nested communities with respect to the order, that is, a solution to Problem 1. Fix i such that $1 \leq i \leq k-1$ and let $b = |V_i|$. Let $X \subseteq V_{i+1} \setminus V_i$ and $Y \subseteq V_i \setminus V_{i-1}$ such that $X = \{v_{b+1}, \ldots, v_{b+|X|}\}$ and $Y = \{v_{b-|Y|+1}, \ldots, v_b\}$. Then $d(X, X \cup V_i) \leq d(Y, V_i)$.*

The only difference between Proposition 1 and Proposition 4 is that in Proposition 4 we require additionally that V_{i+1} starts with X and V_i ends with Y with respect to the order. We want this condition to be redundant, otherwise the given order is suboptimal. For example, consider the adjacency matrix of G given in

Figure 2(a). The given segmentation is optimal with respect to the given order. However if we rearrange the vertices in D_1 and D_2, given in Figure 2(b), then the same segmentation is no longer optimal as X and Y violate Proposition 4. The additional condition in Proposition 4 becomes redundant if V_i ends with the sparsest subset while V_{i+1} starts with densest subset. We will show that the algorithm SortVertices produces an order that satisfies this property *approximately*. The exact formulation of our claim is given as Propositions 5 and 6.

Algorithm 1. SortVertices. Sort vertices of a weighted graph by iteratively removing a vertex with the least weight of adjacent edges.

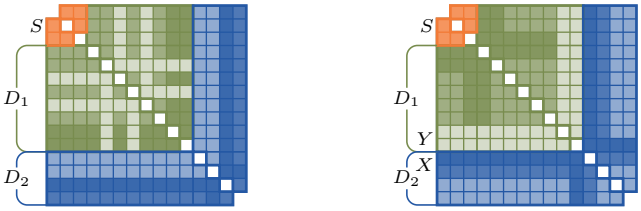
input : weighted graph $G = (V, E, w)$, a set S
output : order on V
$W \leftarrow V \setminus S$;
$o \leftarrow$ empty sequence;
while $|W| > 0$ **do**
  $x \leftarrow \arg\min_{x \in W} d(x, W \cup S)$;
  delete x from W and add x to the beginning of o;
add S in an arbitrary order to the beginning of o;
return o;

Proposition 5. *Consider a weighted graph $G = (V, E, w)$, whose vertices are ordered by algorithm* SortVertices. *Let $1 \le b < c \le |V|$. Let $U = \{v_b, \ldots, v_c\}$ and $W = \{v_1, \ldots, v_c\}$. Let $f = d(v_c, W)$. Then $2f \le d(X, W)$ for any $X \subseteq U$.*

Proof. Note that $s = \sum_{x \in X} w(x, W) = 2w(X) + w(X, W \setminus X) \le 2w(X, W)$. Write $m_x = |c(x, W)|$. Since v_c has the smallest $d(v_c, W)$, we have

$$s = \sum_{x \in X} m_x d(x, W) \ge d(v_c, W) \sum_{x \in X} m_x \ge d(v_c, W)\,|c(X, W)| \quad .$$

Combining the inequalities and dividing by $|c(X, W)|$ proves the result. □



(a) Original order (b) Improved order

Fig. 2. Consequences of Proposition 4. If we reorder the vertices in D_1 and D_2, then an optimal solution with respect to the order may become suboptimal with respect to the improved order.

Proposition 6. *Consider a weighted graph $G = (V, E, w)$, whose vertices are ordered by algorithm* SortVertices. *Let $1 \leq b < c \leq |V|$. Let $U = \{v_b, \ldots, v_c\}$ and $W = \{v_1, \ldots, v_{b-1}\}$. Assume that there is $\alpha \geq 0$ such that for all $v \in U$ it is $\alpha w(v, W) \geq w(v, U)$. Let $f = d(v_b, W)$. Then $(1+\alpha)^2 f \geq d(X, X \cup W)$ for any $X \subseteq U$.*

Proof. Let $A = c(X, W)$ and $B = c(X, X)$. The density of X is bounded by

$$d(X, X \cup W) = \frac{w(A) + w(B)}{|A| + |B|} \leq \frac{w(A) + \alpha w(A)}{|A| + |B|} \leq \frac{(1+\alpha)w(A)}{|A|} = (1+\alpha)d(A)\,.$$

Select $x \in X$ with the highest $d(x, W)$. Then $d(A) \leq d(x, W)$. Let us prove that $d(x, W) \leq (1+\alpha)f$. If $v_b = x$, then we are done. Assume that $v_b \neq x$. Since G is fully-connected, SortVertices always picks the vertex with the lowest weight. Let $Z = \{v_1, \ldots, x\}$. Then $w(x, W) \leq w(x, Z) \leq w(v_b, Z) = w(v_b, W) + w(v_b, U) \leq (1+\alpha)w(v_b, W)$. Since, G is fully-connected $w(y, W) = |W| d(y, W)$ for any $y \in U$. Hence, dividing the inequality gives us $d(x, W) \leq (1+\alpha)f$, which proves the proposition. □

5 Related Work

Finding communities in graphs and social networks is one of the most well-studied topics in graph mining. The amount of literature on the subject is very extensive. This section cannot aspire to cover all the different approaches and aspects of the problem, we only provide a brief overview of the area.

Community Detection. A large part of the related work deals with the problem of partitioning a graph in disjoint clusters or communities. A number of different methodologies have been applied, such as hierarchical approaches [11], methods based on modularity maximization [1, 6, 11, 26], graph-theoretic approaches [8, 9], random-walk methods [21, 24, 28], label-propagation approaches [24], and spectral graph partition [5,15,18,25]. A thorough review on community-detection methods can be found on the survey by Fortunato [10]. We note that this line of work is different than the present paper, since we do not aim at partitioning a graph in disjoint communities.

Overlapping Communities. Researchers in community detection have realized that, in many real situations and real applications, it is meaningful to consider that graph vertices do not belong only to one community. Thus, one asks to partition a graph into overlapping communities. Typical methods here rely on clique percolation [19], extensions to the modularity-based approaches [12, 20], analysis of ego-networks [7], or fuzzy clustering [27]. Again the problem we address in this paper is quite different. First, we find communities centered around a given set of source vertices, and not for the whole graph. Second, the communities output by our algorithm do not have arbitrary overlaps, but they have a specific nested structure.

Centerpiece Subgraphs and Community Search. Perhaps closer to our approach is work related to the centerpiece subgraphs and the community-search

problem [16,22,23]. In this class of problems, a set of source vertices S is given and the goal is to find a subgraph so that S belongs in the subgraph and the subgraph forms a tight community. The quality of the subgraph is measured with various objective functions, such as degree [22], conductance [16], or random-walk-based measures [23]. The difference of these methods with the one presented here is that these methods return only one community, while in this paper we deal with the problem of finding a sequence of nested communities.

In summary, despite the numerous research on the topic of community detection in graphs and social networks, to the best of our knowledge, this is the first paper to address the topic of nested communities with respect to a set of source vertices. Furthermore, our approach offers novel technical ideas, such as providing a solid theoretical analysis that allows to decompose the problem of finding nested communities into two sub-problems: (i) ordering the set of vertices, and (ii) segmenting the graph vertices according to that given order.

6 Experimental Evaluation

We will now provide experimental evidence that our method efficiently discovers meaningful segmentations and that our ordering algorithm outperforms several natural baselines.

Datasets and Experimental Setup. In our experiments we used six datasets, five obtained from Mark Newman's webpage,[2] and a bibliographic dataset obtained from DBLP. The datasets are as follows: *Adjnoun*: adjacency graph of common adjectives and nouns in the novel David Copperfield, by Charles Dickens. *Dolphins*: an undirected social graph of frequent associations between 62 dolphins in a community living off Doubtful Sound, New Zealand. *Karate*: social graph of friendships between 34 members of a karate club at a US university in the 1970s. *Lesmis*: coappearance graph of characters in the novel Les Miserables. *Polblogs*: a directed graph of hyperlinks between weblogs on US politics, recorded in 2005. *DBLP*: coauthorship graph between researchers in computer science. The statistics of these datasets are given in Table 1.

For each dataset and a given source set S, we considered three different weighting schemes: First we run personalized PageRank using the source node with a restart of 0.1. Let $p(v)$ be the PageRank weight of each vertex. Given an edge $e = (v, w)$, we set three different weighting schemes,

$$w_n(e) = \frac{p(v)}{\deg(v)} + \frac{p(w)}{\deg(w)}, \quad w_s(e) = p(v) + p(w), \quad w_m(e) = \min(p(v), p(w)).$$

These weights are selected so that the vertices that are hard to reach with a random walk will have edges with small weights, and hence will be placed in outer communities. For *DBLP*, we weighted the edges during PageRank computation with the number of joint papers, each paper normalized by the number of authors. We use the vertex with the highest degree as a starting set.

[2] `http://www-personal.umich.edu/~mejn/netdata/`

Table 1. Basic statistics of graphs (first two columns) and performance over hops baseline. The third column represents a typical running time while the fourth column represents a typical number of entries during the segmentation. The last three columns represent the normalized score compared to the baseline score $q(\mathcal{H})$.

					performance $q(\mathcal{V})\,/\,q(\mathcal{H})$		
Name	$\lvert V(G)\rvert$	$\lvert E(G)\rvert$	Time	N	w_n	w_s	w_m
Adjnoun	112	425	2ms	84	0.90/0.95	0.88/0.95	0.77/0.94
Dolphins	62	159	1ms	41	0.67/0.80	0.61/0.78	0.57/0.80
Karate	34	78	1ms	21	0.78/0.91	0.76/0.91	0.60/0.93
Lesmis	77	254	2ms	37	0.77/0.93	0.84/0.94	0.62/0.94
Polblogs	1 222	16 714	84ms	872	0.87/0.96	0.95/0.99	0.57/0.96
DBLP	703 193	2 341 362	23s	1 797	0.87/0.99	0.98/1.00	0.45/0.99

Time Complexity. Our first step is to study the running time of our algorithm. We ran our experiments on a laptop equipped with a 1.8 GHz dual-core Intel Core i7 with 4 MB shared L3 cache, and typical running times for each dataset are given in 3rd column of Table 1.[3] Our algorithm is fast: for the largest dataset with 2 million edges, the computation took only 20 seconds. The algorithm consists of 4 steps, computing PageRank, ordering the vertices, grouping the vertices into blocks such that monotonicity condition is guaranteed, and segmenting the groups. The only computationally strenuous step is segmentation which requires quadratic time in the number of blocks. The number of vertices in *DBLP* is over 700 000, however, grouping according to the PAV algorithm leaves only 2 000 blocks, which can be easily segmented. It is possible to select weights in such a way that there will no reduction when grouping vertices, so that finding the optimal segmentation becomes infeasible. However, in such a case, we can always resort to a near-linear approximation optimization algorithm [13].

Comparison to Baseline. A key part in our approach is discovering a good order. Our next step is to compare the order induced by SortVertices against several natural baselines. For the first baseline we group the vertices based on the length of a minimal path from the source. We then compared these communities, say $\mathcal{H}$, to the (same number of) communities obtained with our method. The scores, given in Table 1, show that our approach beats this baseline in every case, which is expected since this naïve baseline does not take into account density. For our next two baselines we order vertices based on vertex degree and PageRank. We then compute community sequences with 2–10 communities from these orders. Typical scores are given in Figure 3. Out of $6 \times 3 \times 9 = 162$ comparisons, SortVertices wins both orders 158 times, ties once (*Karate*, w_m, 3 communities) and loses 3 times to the degree order (*DBLP*, w_n, 3–5 communities).

[3] For the code, see `http://users.ics.aalto.fi/~ntatti/`

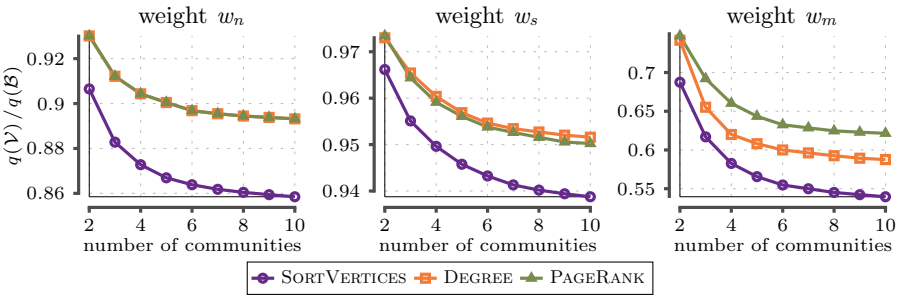
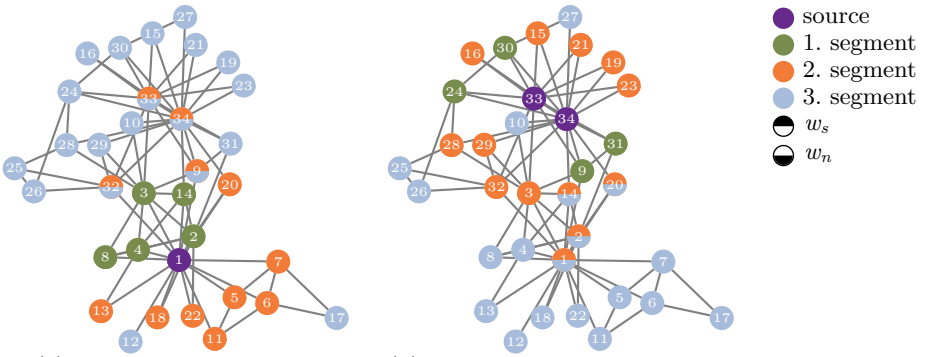


Fig. 3. Quality scores of community sequences based on different orders as a function of number of communities for *Polblogs*. The scores are normalized by the score of a community sequence $\mathcal{B}$ with a single community.

Table 2. Top-3 communities from a sequence of 5 communities for Christos Papadimitriou from *DBLP* set and using w_s

1. segment	D. Johnson	E. Dahlhaus	V. Vianu	G. Gottlob	A. Itai
M. Yannakakis	M. Garey	P. Crescenzi	P. Kanellakis	M. Sideri	A. Schäffer
F. Afrati	R. Karp	P. Seymour	S. Abiteboul	E. Koutsoupias	A. Aho
2. segment	R. Fagin	O. Vornberger	A. Piccolboni	C. Daskalakis	P. Serafini
J. Ullman	**3. segment**	M. Blum	D. Goldman	X. Deng	P. Raghavan
Y. Sagiv	G. Papageorgiou	K. Ross	E. Arkin	P. Goldberg	P. Bernstein
S. Cosmadakis	V. Vazirani	P. Kolaitis	I. Diakonikolas	T. Hadzilacos	



(a) *Karate*, source vertex 1 (b) *Karate*, source vertices 33, 34

Fig. 4. 4 community sequences with 3 communities of *Karate*. Segmentations in Figure 4(a) use 1 as a source and community sequences in Figure 4(b) use 33, 34 as sources. Communities are decoded as colors, the top-half represents w_s, the bottom-half represents w_n.

Examples of Communities. Our final step is to provide examples of discovered communities. In Figure 4 we provide 4 different community sequences with 3 communities using weights w_s and w_n and sources $S = \{1\}$ and $S = \{33, 34\}$.

The inner-most community for 1 contains a near 5-clique. The inner-most community for $33, 34$ contains two 4-cliques. The normalized weight w_n penalizes hubs. This can be seen in Figure 4(a), where hubs 33, 34 move from the outer community to the middle community. Similarly, hub 1 changes communities in Figure 4(b). Finally, we give an example of communities discovered in *DBLP*. Table 6 contains communities discovered around Christos Papadimitriou. Authors in inner communities share many joint papers with Papadimitriou.

7 Concluding Remarks

We considered a problem of discovering nested communities, a sequence of subgraphs such that each community is a more connected subgraph of the next community. We approach the problem by dividing it into two subproblems: discovering the community sequence for a fixed order of vertices, a problem which we can solve efficiently, and discovering an order. We provided a simple heuristic for discovering an order, and provided theoretical and empirical evidence that this order is good.

Discovering nested communities seems to have a lot of potential as it is possible to modify or extend the problem in many ways. We can generalize the problem by not only considering sequences but, for example, trees of communities, where a parent node needs to be a denser subgraph than the child node. Another possible extension is to consider multiple source sets instead of just one.

Acknowledgements. This work was supported by Academy of Finland grant 118653 (ALGODAN).

References

1. Agarwal, G., Kempe, D.: Modularity-maximizing network communities via mathematical programming. European Physics Journal B 66(3) (2008)
2. Ayer, M., Brunk, H., Ewing, G., Reid, W.: An empirical distribution function for sampling with incomplete information. The Annals of Mathematical Statistics 26(4) (1955)
3. Bellman, R.: On the approximation of curves by line segments using dynamic programming. Communications of the ACM 4(6) (1961)
4. Charikar, M.: Greedy approximation algorithms for finding dense components in a graph. In: Jansen, K., Khuller, S. (eds.) APPROX 2000. LNCS, vol. 1913, pp. 84–95. Springer, Heidelberg (2000)
5. Chung, F.R.K.: Spectral Graph Theory. American Mathematical Society (1997)
6. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Physical Review E (2004)
7. Coscia, M., Rossetti, G., Giannotti, F., Pedreschi, D.: DEMON: a local-first discovery method for overlapping communities. In: KDD (2012)
8. Flake, G.W., Lawrence, S., Giles, C.L.: Efficient identification of web communities. In: KDD (2000)

9. Flake, G.W., Lawrence, S., Giles, C.L., Coetzee, F.M.: Self-organization and identification of web communities. Computer 35(3) (2002)
10. Fortunato, S.: Community detection in graphs. Physics Reports, 486 (2010)
11. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. PNAS 99 (2002)
12. Gregory, S.: An algorithm to find overlapping community structure in networks. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 91–102. Springer, Heidelberg (2007)
13. Guha, S., Koudas, N., Shim, K.: Approximation and streaming algorithms for histogram construction problems. ACM TODS 31 (2006)
14. Haiminen, N., Gionis, A.: Unimodal segmentation of sequences. In: ICDM (2004)
15. Karypis, G., Kumar, V.: Multilevel algorithms for multi-constraint graph partitioning. In: CDROM (1998)
16. Koren, Y., North, S.C., Volinsky, C.: Measuring and extracting proximity graphs in networks. TKDD 1(3) (2007)
17. Leskovec, J., Lang, K.J., Dasgupta, A., Mahoney, M.W.: Statistical properties of community structure in large social and information networks. In: WWW (2008)
18. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: NIPS (2001)
19. Palla, G., Derényi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. Nature 435 (2005)
20. Pinney, J., Westhead, D.: Betweenness-based decomposition methods for social and biological networks. In: Interdisciplinary Statistics and Bioinformatics (2006)
21. Pons, P., Latapy, M.: Computing communities in large networks using random walks. Journal of Graph Algorithms Applications 10(2) (2006)
22. Sozio, M., Gionis, A.: The community-search problem and how to plan a successful cocktail party. In: KDD (2010)
23. Tong, H., Faloutsos, C.: Center-piece subgraphs: problem definition and fast solutions. In: KDD (2006)
24. van Dongen, S.: Graph Clustering by Flow Simulation. PhD thesis, University of Utrecht (2000)
25. von Luxburg, U.: A tutorial on spectral clustering. Statistics and Computing 17(4) (2007)
26. White, S., Smyth, P.: A spectral clustering approach to finding communities in graph. In: SDM (2005)
27. Zhang, S., Wang, R.-S., Zhang, X.-S.: Identification of overlapping community structure in complex networks using fuzzy c-means clustering. Physica A (2007)
28. Zhou, H., Lipowsky, R.: Network brownian motion: A new method to measure vertex-vertex proximity and to identify communities and subcommunities. In: Bubak, M., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2004. LNCS, vol. 3038, pp. 1062–1069. Springer, Heidelberg (2004)