Jaccard Median for Ego-network Segmentation

Haodi Zhong[1], Grigorios Loukides[2], Alessio Conte[3], and Solon P. Pissis[4,5]
[1]Xidian University, China
[2]King's College London, United Kingdom
[3]University of Pisa, Italy
[4]CWI, The Netherlands
[5]Vrije Universiteit, The Netherlands
[1]zhonghaodi429@gmail.com, [2]grigorios.loukides@kcl.ac.uk, [3]alessio.conte@unipi.it, and [4]solon.pissis@cwi.nl

***Abstract*—An ego-network is a graph representing the interactions of a node (*ego*) with its neighbors and the interactions among those neighbors. A sequence of ego-networks having the same ego can thus model the evolution of these interactions over time. We introduce the problem of segmenting a sequence of ego-networks into k segments, for any given integer k. Each segment is represented by a summary network, and the goal is to minimize the total loss of representing k segments by k summaries. The problem allows partitioning the sequence into homogeneous segments with respect to the activities or properties of the ego (e.g., to identify time periods when a user acquired different circles of friends in a social network) and to compactly represent each segment with a summary. The main challenge is to construct a summary that represents a collection of ego-networks with minimum loss. To address this challenge, we employ Jaccard Median (JM), a well-known NP-hard problem for summarizing sets, for which, however, no effective and efficient algorithms are known. We develop a series of algorithms for JM offering different effectiveness/efficiency trade-offs: (I) an exact exponential-time algorithm, based on Mixed Integer Linear Programming and (II) exact and approximation polynomial-time algorithms for minimizing an upper bound of the objective function of JM. By building upon these results, we design two algorithms for segmenting a sequence of ego-networks that are effective, as shown experimentally.**

I. Introduction

An ego-network represents the interactions between a focal entity and its neighbors, as well as interactions among those neighbors. It can be modeled as an undirected graph where a node u, called *ego*, must be connected to all other nodes, and there may exist other edges between those nodes. Ego-networks are often constructed from surveys or extracted from larger networks [1]. See Fig. 1 for examples of ego-networks.

A sequence of ego-networks having the same ego can thus naturally model the evolution of the interactions of an entity with its neighbors and the interactions among these neighbors over time. Consider a co-authorship network such as DBLP. A sequence of the ego-networks of an author u represents who u wrote papers with at different times. Also, consider a sequence of ego-networks that are extracted from Twitter and have a popular hashtag u as ego and other hashtags that co-occurred with u as nodes. These ego-networks represent when u co-occurred with some hashtags and if and when these hashtags co-occurred at different times. Similarly, a sequence of ego-networks for a Facebook user u represents when u acquired some friends and if and when the friends of u became friends.

We introduce the problem of partitioning (*segmenting*) a sequence of ego-networks having the same ego u into k contiguous subsequences (*segments*), for any given integer k. Each segment is associated with a network (graph), called *summary*, which represents the ego-networks in the segment compactly. Such a representation of a sequence incurs some loss (*error*) that we seek to minimize. See Fig. 1 for an example illustrating a sequence of ego-networks, its segmentation with $k = 2$, and the summary of each segment.

Solving the ego-network sequence segmentation problem enables us to find k time periods that are as homogeneous as possible with respect to the activities or properties of u and to compactly represent each such time period with a summary. For example, segmenting a sequence of ego-networks of an author u in DBLP allows identifying the time periods in which u worked with different coauthors (e.g., due to their interest in emerging areas) and to obtain a compact representation of u and their coauthors for each time period. Similarly, segmenting a sequence of ego-networks from Facebook allows us to identify time periods when a user's friendships remained stable, as a result of working in the same organization or living in the same city, and also obtain their compact representations.

The ego-network sequence segmentation problem is conceptually similar to the well-known problem of time-series segmentation [2], [3] but introduces two challenges: (I) how to represent a sequence of graphs with a summary; and (II) how to construct an optimal summary efficiently. These challenges do not arise in time-series segmentation, as statistics such as the median or mean can serve as optimal representatives with respect to popular error measures [2]. The problem is also related to anomaly detection [4], [5], [6].

To address challenges I and II, we observe that all ego-networks have the same ego and are aligned (i.e., the correspondence between nodes across networks is known), as they model some known entities (e.g., DBLP authors, Twitter hashtags or Facebook users). Thus, we represent each ego-network by its edge set and propose using the notion of set median (a set with minimum total distance from the sets of a given collection of sets) as a summary that represents the edge sets. Specifically, we construct a summary with respect to Jaccard distance, based on the Jaccard Median (JM) problem [7], [8], [9]. More formally, let X and Y be two sets and $\mathcal{JD}(X,Y) = 1 - \frac{|X \cap Y|}{|X \cup Y|}$ be their Jaccard distance.

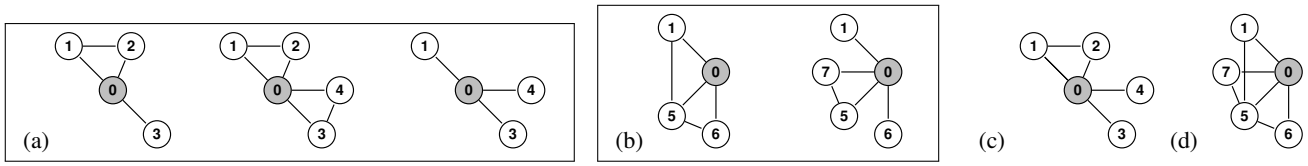


Fig. 1: (a, b) Ego-networks with node 0 as ego. The ego-networks in Fig. 1(a) (respectively, Fig. 1(b)) correspond to the first (respectively, second) segment. (c) Summary of the ego-networks in Fig. 1(a). (d) Summary of the ego-networks in Fig. 1(b).

Given $U = \{u_1, \ldots, u_n\}$ and a collection $\mathcal{C} = \{C_1, \ldots, C_m\}$ of subsets of U, JM asks for a set $M \subseteq U$ such that $\sum_{i \in [m]} \mathcal{JD}(C_i, M)$ is minimized. An optimal JM solution provides us with an edge set that is as close as possible with respect to Jaccard distance to the others in the collection, thus a good summary. For example, a solution to JM with the collection of edge sets of the ego-networks in Fig. 1(a) as input is shown in Fig. 1(c).

Jaccard distance is suitable for ego-network segmentation because: (I) it has a clear interpretation as the proportion of edges that are in only one input edge set with respect to the total number of edges appearing in either edge set; (II) it is effective at capturing changes to the underlying graph structure around a node over time [10]; and (III) it is computable in linear time in the total size of the two input edge sets.

Unfortunately, however, JM is NP-hard to compute *exactly* and (assuming P≠NP) cannot have a *fully* polynomial-time approximation scheme (FPTAS) [9]. In addition, the existing approximation algorithms for JM are not satisfactory in practice. Two algorithms are known [9]: a 2-approximation algorithm and a PTAS $(1+\epsilon)$-approximation algorithm, for any $\epsilon > 0$. The first is a simple $O(nm^2)$-time algorithm, which returns the best element of $\mathcal{C}$ with respect to Jaccard distance as a solution. This algorithm is ineffective, as shown in our experiments. The second algorithm is inefficient as it takes $(nm)^{\frac{1}{\epsilon^{O(1)}}}$ time, which implies that, even for $\epsilon = 1$ needed to obtain a 2-approximation, it takes $\Omega(nm)$ time.

Contributions. We introduce the problem of segmenting a sequence of ego-networks and solve it via proposing practical algorithms for JM. In summary, our contributions are:

1. We develop an exact algorithm for JM, based on Mixed Integer Linear Programming (MILP). Thus, using any specialized solver, we can find a single optimal solution and also enumerate all optimal solutions. In accordance with the NP-hardness of JM, the algorithm takes in general exponential time in the size of the input; it is however useful for evaluating our other algorithms on small instances.

2. We alleviate the computational difficulty of JM by plugging in the Sørensen-Dice coefficient [11], a well-known similarity measure for sets, instead of the Jaccard distance. Specifically, we provide an upper bound for the objective function of JM based on the Sørensen-Dice coefficient and develop EXACT-SC_{UB}, an exact $O(n^3m)$-time algorithm that minimizes the bound. We also develop two more efficient variants of EXACT-SC_{UB}. The first constructs an exact solution of fixed size ρ in $O(nm\rho)$ time; it may be useful when one wants to control the summary size (e.g., for compactly representing a sequence based on the segment summaries). The second variant constructs an exact solution in $O(n^2m)$ time, when all ego-networks have the same size. If the latter does not hold, it is still a very effective approximation algorithm. Both EXACT-SC_{UB} and its variants are orders of magnitude faster than the MILP algorithm and find near-optimal solutions to the JM problem, as shown in our experiments.

3. We build on the above results to design two segmentation algorithms: one exact based on dynamic programming [12], [2]; and another based on a top-down heuristic [2]. Both algorithms can use any of our algorithms for the JM problem as a subroutine to construct the summaries.

4. We present experiments showing that our algorithms: (I) find optimal or near-optimal solutions to the JM problem, substantially outperforming the 2-approximation algorithm of [9]; and (II) are much more effective in ego-network segmentation than several state-of-the-art methods [13], [4], [5], [6].

II. Background and Problem Statements

Preliminaries. An *ego-network* is a triple $G = (u, V, E)$, such that $V = \{u, v_1, \ldots, v_{|V|-1}\}$ and $\{(u, v_1), \ldots, (u, v_{|V|-1})\} \subseteq E \subseteq V \times V$. That is, every node $v_i \neq u$, $i \in [1, |V|-1]$, is connected to u and can potentially be connected to other nodes. Let X and Y be two sets. Based on the *inclusion/exclusion principle*, it holds that $|X \cup Y| = |X| + |Y| - |X \cap Y|$. The *Jaccard coefficient* of X and Y is defined as $\mathcal{JC}(X,Y) = \frac{|X \cap Y|}{|X \cup Y|}$ and the *Jaccard distance* is defined as $\mathcal{JD}(X,Y) = 1 - \frac{|X \cap Y|}{|X \cup Y|}$. Another popular similarity measure for sets is the *Sørensen-Dice coefficient* [11]. For sets X and Y, it is defined as $\mathcal{SDC}(X,Y) = 2\frac{|X \cap Y|}{|X|+|Y|}$ and can be easily rewritten as $\frac{1-\mathcal{JD}(X,Y)}{1-\mathcal{JD}(X,Y)/2}$. Let U be a universe of elements and 2^U be its power set. A function $f : 2^U \to \mathbb{R}$ is *monotone*, if $f(X) \leq f(Y)$ for all subsets $X \subseteq Y \subseteq U$. A function f that is not monotone is referred to as *non-monotone*.

Segmentation. A *k-segmentation* of a sequence $\mathcal{E}_{1,m} = E_1, \ldots, E_m$ of sets is a partition of $\mathcal{E}_{1,m}$ into k contiguous subsequences (segments) $\mathcal{E}_{1,j_1}, \ldots, \mathcal{E}_{j_{k-1}+1,m}$, where the subscripts in a subsequence denote the indices of the first and last set in the subsequence. Let $E_{1,j_1}, \ldots, E_{j_{k-1}+1,m}$ be k feasible JM solutions (summaries) for $\mathcal{E}_{1,j_1}, \ldots, \mathcal{E}_{j_{k-1}+1,m}$. The *error* of this k-segmentation is then calculated as $\sum_{i=1}^{j_1} \mathcal{JD}(E_i, E_{1,j_1}) + \ldots + \sum_{i=j_{k-1}+1}^{m} \mathcal{JD}(E_i, E_{j_{k-1}+1,m})$. We next define the *Set Sequence Segmentation* (SSS) problem.

Problem 1 (Set Sequence Segmentation (SSS))**.** *Given a sequence $\mathcal{E}_{1,m} = E_1, \ldots, E_m$ of sets and an integer $k > 0$, find a k-segmentation of $\mathcal{E}_{1,m}$ minimizing the error.*

Since Problem 1 (SSS) with $k = 1$ is essentially JM, it is NP-hard and there is no FPTAS for it (assuming P≠NP) [9].

We now consider a collection of ego-networks that have the same ego. As the ego-networks are aligned, we view each ego-network merely as its set of edges. We can thus provide the edge sets $E_1, \ldots, E_m$ of the ego-networks as input to JM to construct a set $E_{[m]}$ of edges that has minimum total Jaccard distance from the edge sets. For brevity, we may denote $|\cup_{i\in[m]} E_i|$ by n and $\sum_{i\in[m]} |E_i|$ by N.

We refer to $E_{[m]}$ as the *summary* of $E_1, \ldots, E_m$ and to the JM problem that takes as input the edge sets $E_1, \ldots, E_m$ as *Summary Construction* (SC). Similarly, we refer to the SSS that takes as input a sequence of edge sets $E_1, \ldots, E_m$ as *Ego-network Sequence Segmentation* (ESS). Note that the order of the input edge sets plays no role in SC, but it is crucial to be maintained in ESS, as the segmentation will be different should the order change.

III. SC as a Mixed Integer Linear Program

The objective function in the SC problem can be written as $\sum_{i\in[m]} \mathcal{JD}(E_i, E_{[m]}) = \sum_{i\in[m]}(1-\frac{|E_i\cap E_{[m]}|}{|E_i\cup E_{[m]}|})$. Thus, we can minimize it by maximizing $\sum_{i\in[m]} \frac{|E_i\cap E_{[m]}|}{|E_i\cup E_{[m]}|}$. We achieve this by a novel Mixed Integer Linear Program (MILP) formulation.

Recall that $n = |\cup_{i\in[m]} E_i|$. We associate each edge $e_j \in \cup_{i\in[m]} E_i$ with a binary variable $x_j, j \in [n]$, which is equal to 1 if $e_j \in E_{[m]}$ and 0, otherwise. We also consider a coefficient a_{ij} which is equal to 1 if $e_j \in E_i$ and 0, otherwise. Thus, we can express $|E_{[m]}|$ as $\sum_{j=1}^{n} x_j$ and $|E_i \cap E_{[m]}|$ as $\sum_{j=1}^{n} a_{ij}x_j$. Furthermore, based on the inclusion/exclusion principle, we can write $|E_i \cup E_{[m]}| = |E_i| + |E_{[m]}| - |E_i \cap E_{[m]}|$ and express it as $|E_i| + \sum_{j=1}^{n} x_j - \sum_{j=1}^{n} a_{ij}x_j = |E_i| + \sum_{j=1}^{n}(1-a_{ij})x_j$. Therefore, $\sum_{i\in[m]} \frac{|E_i\cap E_{[m]}|}{|E_i\cup E_{[m]}|}$ is maximized by solving the following fractional 0-1 program:

$$\max \quad \sum_{i=1}^{m} \frac{\sum_{j=1}^{n} a_{ij}x_j}{|E_i| + \sum_{j=1}^{n}(1-a_{ij})x_j} \tag{1a}$$

$$\text{s.t.} \quad x_j \in \{0,1\}, \quad j \in [n]. \tag{1b}$$

Fractional 0-1 programs are in general very difficult to solve directly due to their nonlinear objective functions and the discrete nature of their decision variables. However, it is fortunately possible to formulate our fractional program as an MILP, based on the general methodology of [14]. We do this in two steps. First, we use a substitution variable $y_i = \frac{1}{|E_i| + \sum_{j=1}^{n}(1-a_{ij})x_j}$, for each $i \in [m]$. Since $y_i \in [\frac{1}{|E_i|+n}, \frac{1}{|E_i|}]$, for each $i \in [m]$, the substitution leads to the following nonlinear reformulation of the fractional program:

$$\max \quad \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}y_i x_j \tag{2a}$$

$$\text{s.t.} \quad |E_i|y_i + \sum_{j=1}^{n}(1-a_{ij})y_i x_j - 1 = 0, i \in [m] \tag{2b}$$

$$x_j \in \{0,1\} \quad , j \in [n] \tag{2c}$$

$$y_i \in [\frac{1}{|E_i|+n}, \frac{1}{|E_i|}] \quad , i \in [m], \tag{2d}$$

where Eq. 2b is introduced to enforce the substitution. In the second step, we linearize this program by setting $z_{ij} = y_i x_j$, for all $i \in [m]$ and $j \in [n]$. By doing so, we obtain the following MILP formulation of SC:

$$\max \quad \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}z_{ij} \tag{3a}$$

$$\text{s.t.} \quad |E_i|y_i + \sum_{j=1}^{n}(1-a_{ij})z_{ij} - 1 = 0 \ , i \in [m] \tag{3b}$$

$$z_{ij} \le \frac{1}{|E_i|}x_j \quad , \ i \in [m], j \in [n] \tag{3c}$$

$$z_{ij} \le y_i + \frac{1}{|E_i|+n}(x_j - 1) \quad , \ i \in [m], j \in [n] \tag{3d}$$

$$z_{ij} \ge \frac{1}{|E_i|+n}x_j \quad , \ i \in [m], j \in [n] \tag{3e}$$

$$z_{ij} \ge y_i + \frac{1}{|E_i|}(x_j - 1) \quad , \ i \in [m], j \in [n] \tag{3f}$$

$$x_j \in \{0,1\} \quad , \ j \in [n] \tag{3g}$$

$$y_i \in [\frac{1}{|E_i|+n}, \frac{1}{|E_i|}] \quad , \ i \in [m]. \tag{3h}$$

The constraints in Eqs. 3c to 3f are introduced to enforce $z_{ij} = y_i x_j$, for all $i \in [m]$ and $j \in [n]$. To see this, let $x_j = 0$ for some $j \in [n]$. Then, we get $z_{ij} \le 0$ from Eq. 3c and $z_{ij} \ge 0$ from Eq. 3e, which implies $z_{ij} = y_i x_j = y_i \cdot 0 = 0$. Now, let $x_j = 1$. Then, we get $z_{ij} \le \min(\frac{1}{|E_i|}, y_i) = y_i$ from Eqs. 3c and 3d and $z_{ij} \ge \max(\frac{1}{|E_i|+n}, y_i) = y_i$ from Eqs. 3e and 3f, which imply $z_{ij} = y_i x_j = y_i \cdot 1 = y_i$. Similarly, we can observe that $z_{ij} = y_i x_j$ for each y_i satisfying Eq. 3h.

Finally, given an optimal solution of the MILP, we obtain an optimal summary $E_{[m]}$ by adding, for each $j \in [n]$ such that $x_j = 1$, an edge e_j into $E_{[m]}$. The resulting algorithm is referred to as EXACT-SC. We obtain the following.

Theorem 1. EXACT-SC *solves the SC problem exactly.*

IV. Greedy Algorithms for an Upper Bound of $\mathcal{JD}$

The computational difficulty of the SC problem is due to the use of Jaccard distance. To alleviate this difficulty, we consider a variant of the SC problem where an upper bound of Jaccard distance is used instead. The upper bound we consider is based on the Sørensen-Dice coefficient, as can be seen in the following lemma (the proof is deferred to the full version):

Lemma 1 ($\mathcal{JD}$ Upper Bound)**.** *For any collection $E_1, \ldots, E_m$ and any summary $E_{[m]}$, it holds that $\sum_{i\in[m]} \mathcal{JD}(E_i, E_{[m]}) \le \sum_{i\in[m]}(1 - \frac{1}{2}\mathcal{SDC}(E_i, E_{[m]}))$.*

We define $\mathcal{JD}_{UB}(E_i, E_{[m]}) = 1 - \frac{1}{2}\mathcal{SDC}(E_i, E_{[m]}) = 1 - \frac{|E_i\cap E_{[m]}|}{|E_i|+|E_{[m]}|}$ and refer to the variant of SC using $\mathcal{JD}_{UB}$ as SC_{UB} (SC Upper Bound). We also observe the following (the proof is deferred to the full version):

Lemma 2. *Let $E_1, \ldots, E_m$ be a collection, $U = \cup_{i\in[n]} E_i$, and $E_{[m]} \subseteq U$. The function $\sum_{i\in[m]} \mathcal{JD}_{UB}(E_i, E_{[m]})$ is non-monotone with respect to $E_{[m]}$.*

An exact algorithm for SC_{UB}. Notably, we show that unlike the JM problem, SC_{UB} can be solved in polynomial time:

Theorem 2. *SC_{UB} can be solved in $O(n^3 m)$ time.*

Specifically, we provide EXACT-SC_{UB}. This algorithm

applies an algorithm for SC_{UB} with *fixed solution size* ρ as a subroutine for all $\rho \in [0, n]$, and it returns as solution a set E^ρ that minimizes $\sum_{i\in[m]}(1 - \frac{|E_i \cap E^\rho|}{|E_i|+\rho})$ over all ρ's.

The subroutine algorithm is called FIXED-SIZE-GREEDY-SC_{UB}, and it performs ρ iterations for a given ρ. It starts with $E^1 = E^0 = \{\}$ and, in each iteration $j \in [1, \rho]$, it adds into a set E^j that contains all previously added edges, an edge $e \in \cup_{i\in[m]} E_i \setminus E^j$ minimizing $\sum_{i\in[m]}(1 - \frac{|E_i \cap \{E^j \cup \{e\}\}|}{|E_i|+\rho})$. When $j = \rho$, it returns the set E^j. Note that the value of the output $\sum_{i\in[m]}(1 - \frac{|E_i \cap E^\rho|}{|E_i|+\rho})$ differs from the upper bound $\sum_{i\in[m]} \mathcal{JD}_{UB}(E_i, E_{[m]}) = \sum_{i\in[m]}(1 - \frac{|E_i \cap E_{[m]}|}{|E_i|+|E_{[m]}|})$ in that $E_{[m]}$ is replaced by E^ρ, which has now fixed size ρ.

FIXED-SIZE-GREEDY-SC_{UB} solves exactly a variant of SC_{UB}, called SC_{UB}-FIXED, which asks for a solution of fixed size ρ, for some given ρ. We prove this in Lemma 5 by means of proving Lemmas 3 and 4 (the proofs of Lemmas 3, 4, and 5 are deferred to the full version of the paper).

Lemma 3 (Greedy choice)**.** *Let e be the element contained in E^1 that is constructed by* FIXED-SIZE-GREEDY-SC_{UB}. *There is an optimal solution to SC_{UB}-FIXED that contains e.*

Lemma 4 (Optimal Substructure)**.** *Let $E^\rho_{[m]}$ be any optimal solution to SC_{UB}-FIXED for some $\rho \in [0, n-1]$. There is an edge $e \in \cup_{i\in[m]} E_i \setminus E^\rho_{[m]}$ that can be added into $E^\rho_{[m]}$ to construct an optimal solution to SC_{UB}-FIXED of size $\rho + 1$.*

Lemma 5 (Induction)**.** FIXED-SIZE-GREEDY-SC_{UB} *finds an optimal solution to SC_{UB}-FIXED.*

FIXED-SIZE-GREEDY-SC_{UB} takes $O(nm\rho)$ time. To achieve this time complexity, we compute $\sum_{i\in[m]}(1 - \frac{|E_i \cap \{E^j \cup \{e\}\}|}{|E_i|+\rho})$ incrementally. That is, we exploit the following: (I) $|E_i \cap \{E^j \cup \{e\}\}| = |E_i \cap E^j| + |E_i \cap \{e\}|$, which holds by the distributivity of $\cup$ over $\cap$ and the inclusion/exclusion principle, and (II) that $|E_i \cap \{e\}|$ with all E_i's can be computed in $O(m)$ time, after building a perfect hashtable for every E_i in total $O(N)$ time in which we search e. Since FIXED-SIZE-GREEDY-SC_{UB} performs ρ iterations, it needs $O(N + nm\rho) = O(nm + nm\rho) = O(nm\rho)$ time.

The proof of Theorem 2 follows from Lemma 5 and from that EXACT-SC_{UB} applies FIXED-SIZE-GREEDY-SC_{UB} for each ρ which costs $O(nm(1 + \ldots + n)) = O(n^3 m)$.

A faster–and sometimes optimal–approximation algorithm for SC_{UB}. We show GREEDY-SC_{UB}, a greedy algorithm for SC_{UB} that is polynomially faster than EXACT-SC_{UB}. GREEDY-SC_{UB} is similar to FIXED-SIZE-GREEDY-SC_{UB}, but it: (I) selects e based on the upper bound $\sum_{i\in[m]} \mathcal{JD}_{UB}(E_i, E^j \cup \{e\}) = \sum_{i\in[m]}(1 - \frac{|E_i \cap \{E^j \cup \{e\}\}|}{|E_i|+|E^j|+1})$ instead of $\sum_{i\in[m]}(1 - \frac{|E_i \cap \{E^j \cup \{e\}\}|}{|E_i|+\rho})$; and (II) selects $E_{[m]}$ as the E^j with minimum $\sum_{i\in[m]} \mathcal{JD}_{UB}(E_i, E^j)$ over all $j \in [n]$; this is needed due to non-monotonicity (see Lemma 2).

Thus, GREEDY-SC_{UB} makes the greedy choice optimally for the current size of E^j (after the addition of e), instead of making the optimal choice for a target size ρ. The advantage of this strategy is that each intermediate E^j is immediately considered in the tentative solution for size $|E^j|$, and it is not necessary to run the whole process, for each target size.

Next, we prove the approximation guarantee of GREEDY-SC_{UB} and that it is optimal when all E_i's have equal size.

Let $s_m = \min_{i\in[m]} |E_i|$ and $s_M = \max_{i\in[m]} |E_i|$ be the smallest and largest size of an input set E_i. We show that GREEDY-SC_{UB} is guaranteed to return a good approximation of an optimal solution whenever these two values are close.

Lemma 6. *Let E^j, with $|E^j| = j$, be the set of edges selected by* GREEDY-SC_{UB} *up to step j, for any $j \in [n]$, and E^* be an optimal solution of fixed size j of SC_{UB}, i.e., the set of size j minimizing $\sum_{i\in[m]}(1 - \frac{|E_i \cap E^*|}{|E_i|+j})$. For some $1 \le \phi \le \frac{s_M^2}{s_m^2}$, we have that $\sum_{i\in[m]}(1 - \frac{|E_i \cap E^j|}{|E_i|+j} \cdot \phi) \le \sum_{i\in[m]}(1 - \frac{|E_i \cap E^*|}{|E_i|+j})$.*

Proof. Observe that $\sum_{i\in[m]}(1 - \frac{|E_i \cap E^*|}{|E_i|+j}) = m - \sum_{i\in[m]} \frac{|E_i \cap E^*|}{|E_i|+j}$. The rest of the proof shows that $\sum_{i\in[m]} \frac{|E_i \cap E^*|}{|E_i|+j} \le \sum_{i\in[m]}(\frac{|E_i \cap E^j|}{|E_i|+j}) \cdot \phi$, which implies $m - \sum_{i\in[m]} \frac{|E_i \cap E^*|}{|E_i|+j} \ge m - \left(\sum_{i\in[m]} \frac{|E_i \cap E^j|}{|E_i|+j}\right) \cdot \phi$, proving the lemma.

For an integer $h \in [j]$ and any edge $e \in \cup_{i\in[m]} E_i$, we define $\mathtt{C}_h(e) = \sum_{i\in[m]} \frac{|E_i \cap \{e\}|}{|E_i|+h}$, the "contribution" of e to the score of a solution of size h. We can now easily see that $\sum_{e\in E^*} \mathtt{C}_j(e) = \sum_{i\in[m]} \frac{|E_i \cap E^*|}{|E_i|+j}$. This lets us evaluate edges independently of each other and even predict what contribution they will bring to a solution of some specific size.

Note, we can turn $\mathtt{C}_h(e) = \sum_{i\in[m]} \frac{|E_i \cap \{e\}|}{|E_i|+h}$ into $\mathtt{C}_j(e)$ by taking each $\frac{|E_i \cap \{e\}|}{|E_i|+h}$ and multiplying it by $\frac{|E_i|+h}{|E_i|+j}$. Since $\frac{s_m+h}{s_M+j} \le \frac{|E_i|+h}{|E_i|+j} \le \frac{s_M+h}{s_m+j}$, it follows: $\mathtt{C}_j(e) \ge \mathtt{C}_h(e) \cdot \frac{s_m+h}{s_M+j}$ and $\mathtt{C}_j(e) \le \mathtt{C}_h(e) \cdot \frac{s_M+h}{s_m+j}$.

At a generic step $h \le j$, let e_h be the edge maximizing $\mathtt{C}_h(e)$ and e_{opt} be the edge maximizing $\mathtt{C}_j(e)$, among the edges not previously selected. We can observe that GREEDY-SC_{UB} will select e_h, while FIXED-SIZE-GREEDY-SC_{UB} would instead select e_{opt}.

From the above and from the fact that $\mathtt{C}_h(e_{opt}) \le \mathtt{C}_h(e_h)$ by the definition of e_h, we have $\mathtt{C}_j(e_{opt}) \le \mathtt{C}_h(e_{opt}) \cdot \frac{s_M+h}{s_m+j} \le \mathtt{C}_h(e_h) \cdot \frac{s_M+h}{s_m+j} \le \left(\mathtt{C}_j(e_h) \cdot \frac{s_M+j}{s_m+h}\right) \cdot \frac{s_M+h}{s_m+j}$. This bounds the difference of contributions between the selection made by the two algorithms, and although the bound depends on the step h, we can upper bound the ratio since $1 \le \frac{(s_M+j)(s_M+h)}{(s_m+h)(s_m+j)} \le \phi = \frac{(s_M+j)s_M}{s_m(s_m+j)} \le \frac{s_M^2}{s_m^2}$, obtaining $\mathtt{C}_j(e_{opt}) \le \mathtt{C}_j(e_h) \cdot \phi$.

This completes the proof, as E^j consists of edges each maximizing $\mathtt{C}_h(e)$ at some step $h \le j$, whereas E^* consists instead of the edges maximizing $\mathtt{C}_j(e)$, which in turn gives $m - \sum_{e\in E^*} \mathtt{C}_j(e) \ge m - \sum_{e\in E^j} \mathtt{C}_j(e) \cdot \phi$. □

The ratio $\frac{s_M^2}{s_m^2}$ is not necessarily small, but the analysis is pessimistic: GREEDY-SC_{UB} is near-optimal in practice. In fact, our experiments in Section VI show that GREEDY-SC_{UB} obtains almost the same results as EXACT-SC_{UB} while being up to orders of magnitude more efficient.

Furthermore, we can now easily show a class of instances where GREEDY-SC_{UB} is optimal.

Corollary 6.1. GREEDY-SC_{UB} *is optimal when* $s_m = s_M$.

Proof. If $s_m = s_M$ then, for any $j \in [n]$, $\phi = \frac{(s_M+j)s_M}{s_m(s_m+j)} = 1$. Thus, $\sum_{i\in[m]}(1 - \frac{|E_i \cap E^j|}{|E_i|+j} \cdot \phi) = \sum_{i\in[m]}(1 - \frac{|E_i \cap E^*|}{|E_i|+j})$, where E^j is the set of edges selected up to step j by GREEDY-SC_{UB}, and E^* is an optimal solution of size j. An optimal solution of arbitrary size is thus found when GREEDY-SC_{UB} selects an E^j minimizing the summation. □

It is easy to see that GREEDY-SC_{UB} takes $O(n^2m)$ time. Thus, we obtain the following result:

Theorem 3. *SC_{UB} can be solved in $O(n^2m)$ time when $|E_1| = \ldots = |E_m|$.*

V. Ego-network Sequence Segmentation

The error in the ESS problem is incurred by representing each segment by its summary and is expressed as a sum of the errors of the k segments. Since EXACT-SC constructs a minimum-error (i.e., optimal) summary for any segment, we can solve ESS by considering all k-segmentations and selecting the one with minimum total error. For this task, we adapt a classic dynamic programming algorithm [12], [2].

Let $\mathbf{E}^*(\mathcal{E}_{1,m}, k)$ be the error of an optimal solution to ESS (see Section II). Our algorithm, EXACT-ESS, solves ESS for a sequence $\mathcal{E}_{1,m} = E_1, \ldots, E_m$ of ego-networks by recursively applying Eq. 4 and finding an optimal k-segmentation of $\mathcal{E}_{1,m}$ through standard backtracking [2]:

$$\mathbf{E}^*(\mathcal{E}_{1,m}, k) = \min_{0<j<m} \left\{\mathbf{E}^*(\mathcal{E}_{1,j}, k-1) + \mathbf{E}^*(\mathcal{E}_{j+1,m}, 1)\right\}. \quad (4)$$

$\mathbf{E}^*(\mathcal{E}_{1,j}, k-1)$ is the error of optimally segmenting $\mathcal{E}_{1,j}$, the sequence comprised of the first j ego-networks, into $k-1$ segments; and $\mathbf{E}^*(\mathcal{E}_{j+1,m}, 1)$ is the error of representing the single-segment sequence $\mathcal{E}_{j+1,m}$ by its optimal summary, constructed by EXACT-SC.

Eq. 4 requires invoking EXACT-SC $O(m^2k)$ times. To trade-off accuracy for speed, we can construct the summaries using any of our other algorithms. Yet, constructing a k-segmentation needs $\Omega(m^2)$ time, which is impractical.

Therefore, we partition the sequence of ego-networks $\mathcal{E}_{1,m}$ by adapting a top-down greedy algorithm [2] to work with a sequence of ego-networks and any of our algorithms for constructing a summary. The top-down algorithm starts with the entire sequence $\mathcal{E}_{1,m}$ and applies Eq. 4 with $k = 2$. Let j_1 be the split point found by this process. Then, it applies Eq. 4 with $k = 2$ to $\mathcal{E}_{1,j_1}$ and to $\mathcal{E}_{j_1+1,m}$. This results in two candidate split points, out of which the one with the lowest error is selected as the second split point, j_2, of $\mathcal{E}_{1,m}$. The process continues in the same way, until $k-1$ split points, defining a k-segmentation of $\mathcal{E}_{1,m}$, are selected, and it requires invoking a summary construction algorithm $O(mk)$ times.

VI. Experimental Evaluation

Datasets. We extracted 5 ego-network sequences from DBLP, a real dataset modeling temporal graphs, obtained from https://projects.csail.mit.edu/dnd/DBLP/. To extract an ego-network sequence, we: (I) selected a node u as ego; (II) extracted, as the ego-network of u, the subgraph induced by all edges adjacent to u and edges between its neighbors with timestamps within a time window of certain length; and (III) extracted the next ego-network by sliding the time window by a sliding interval of 4 years. The sliding window model leads to ego-networks with common edges which make segmentation challenging. We defined a segment as a maximal contiguous subsequence of ego-networks sharing at least one edge; the ego-networks in the end of a segment share many edges with those in the beginning of the next one. Each segment was assigned a different ground truth label. The average number m of egonetworks over the 5 extracted sequences is 29.8, the average number n of distinct edges is 2,662, the average total number N of edges is 9,780.8, and the number k of segments is in $[5, 8]$.

Experimental Setup. We evaluated the effectiveness of our dynamic programming (DP) and top-down (T) algorithm, paired with our summary construction algorithms EXACT-SC (ESC), EXACT-SC_{UB} (ESC_{UB}), and GREEDY-SC_{UB} (GSC_{UB}), in ego-network segmentation. We compared against SnapNets [13], the only method for segmenting a sequence of graphs, and state-of-the-art anomaly detection methods: COPOD [4], ECOD [5], and ROD [6]. The latter methods get as input a sequence of real numbers and identify a specified number of anomalies. We constructed an input sequence comprised of: (I) $\mathcal{JD}(E_i, E_{i+1})$ for each $i \in [1, m-1]$; or (II) the L_2 distance $L_2(\text{FV}(E_i), \text{FV}(E_{i+1}))$ for each $i \in [1, m-1]$, where FV() outputs a vector comprised of all features used in [13]. A subscript D (respectively, F) for an anomaly detection method (e.g., COPOD_D) denotes that the input sequence is based on Jaccard distance (respectively, L_2 distance). All parameters of the methods we compared against were set to their default values from [13], [4], [5], [6].

We evaluated the quality of a segmentation by comparing it to the ground truth segmentation using Accuracy (ACC) [15] and macro-F_1 [16]. These measures take values in $[0, 1]$ with larger values indicating a result closer to the ground truth. However, they require a segmentation to have the same number of segments as the ground truth segmentation. To compare against SnapNets, which does not take the desired number of segments as input, we used the Average Sum of Pairwise L_2 distance of NetLSD signatures $\frac{1}{k}\sum_{S\in\mathcal{S}}\sum_{E,E'\in S}\text{NetLSD}(E, E')$ (ASPLN), where E, E' are the edge sets of two ego-networks in a segment S of a k-segmentation $\mathcal{S}$. NetLSD creates signatures representing two graphs, so that close signatures correspond to similar graphs, and it was configured as in [17].

All experiments ran on an Intel i9@3.70GHz with 64 GB RAM. We implemented our methods in Python and used Gurobi 9.5.1 for the MILP instances. See https://rebrand.ly/egoseg for our code and datasets. We used publicly available Python implementations of all other methods.

SC. We selected a contiguous subsequence of length m from

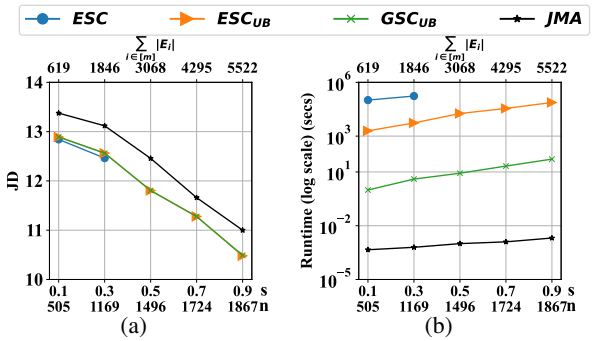


Fig. 2: DBLP: (a) JD and (b) runtime (secs) for varying s and $m = 15$. ESC did not terminate within 48 hours for large s.

TABLE I: (a) ACC and macro-F_1. ACC and macro-F_1 cannot be used with SnapNets as it does not get k as input. (b) Runtime (secs) averaged over all sequences of DBLP.

Methods	ACC	macro-F_1
COPOD$_D$	0.55	0.48
COPOD$_F$	0.47	0.42
T-ESC$_{\text{UB}}$	0.78	0.76
DP-GSC$_{\text{UB}}$	0.72	0.71
T-GSC$_{\text{UB}}$	0.72	0.71

(a)

Methods	Runtime (secs)
COPOD$_D$	0.02
COPOD$_F$	658.02
T-ESC$_{\text{UB}}$	593, 514
DP-GSC$_{\text{UB}}$	10, 101
T-GSC$_{\text{UB}}$	3, 302

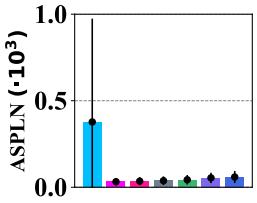
(b)

the first ego-network sequence of DBLP and sampled $s \cdot 100\%$ of edges from each ego-network, so that an E_i for s contains all edges in E_i for $s' < s$. The selected subsequence contained moderately similar ego-networks to be aligned with the goal of constructing an informative summary. Fig. 2a and 2b shows the results in terms of JD and runtime, for varying s, respectively. All our algorithms outperformed JMA in terms of effectiveness. As expected, ESC produced the best result since it is exact for SC; and, notably, our other algorithms produced near-optimal solutions. In terms of runtime, the results are in line with the time complexity analyses. ESC is the slowest, followed by ESC$_{\text{UB}}$ and GSC$_{\text{UB}}$.

Segmentation. We first compared our algorithms to the methods that can segment a sequence of ego-networks in a given number k of segments, which was set to the number of ground truth segments. These are the anomaly detection methods COPOD, ECOD, ROD and a baseline combining our top-down algorithm with JMA. The latter did not perform well and we omit its results. Among the former methods, COPOD performed the best, thus we omit the others. As can be seen in Table Ia, all our methods substantially outperformed both variations of COPOD. COPOD$_D$ outperformed COPOD$_F$, which implies that the features proposed by [13] for graph segmentation may not be suitable for ego-network segmentation. EXACT-ESS and DP-ESC$_{\text{UB}}$ did not terminate within 48 hours (expected).

We also compared our algorithms against SnapNets, which automatically determines the number of segments. To do this, we used as k in our algorithms the number of segments in the solutions of SnapNets. The results in Fig. 3a show that all our methods significantly outperformed SnapNets with respect to ASPLN, which is very encouraging.

In terms of efficiency, Table Ib shows the runtime of all



(a)

Methods	Runtime (secs)
SnapNets	659
T-ESC$_{\text{UB}}$	596, 174
DP-GSC$_{\text{UB}}$	10, 101
T-GSC$_{\text{UB}}$	3, 454

(b)

Fig. 3: (a) ASPLN and (b) runtime (secs) averaged over all sequences of DBLP. Our omitted methods did not terminate within 48 hours.

algorithms for the experiment of Table Ia and Fig. 3b for the experiment of Fig. 3a. Our algorithms were slower albeit more effective than COPOD$_D$, COPOD$_F$, and SnapNets; their runtime is in line with their time complexities.

VII. Final Remarks

We introduced the problem of segmenting a sequence of ego-networks based on the JM problem. It would be interesting to design faster methods and optimize the performance of the current ones exploiting properties of the similarity function.

Acknowledgments

HZ is supported by a CSC scholarship, GL by the Leverhulme Trust RPG-2019-399, and AC partially supported by MUR, PRIN Project n. 20174LF3T8 AHeAD. We would like to thank Prof. Roberto Grossi for useful discussions.

References

[1] R. A. Hanneman and M. Riddle, *Introduction to social network methods*, 2005.
[2] E. Terzi and P. Tsaparas, "Efficient algorithms for sequence segmentation," in *SDM*, 2006.
[3] N. Tatti, "Strongly polynomial efficient approximation scheme for segmentation," *IPL*, 2019.
[4] Z. Li, Y. Zhao, N. Botta, C. Ionescu, and X. Hu, "COPOD: copula-based outlier detection," in *ICDM*, 2020.
[5] Z. Li, Y. Zhao, X. Hu, N. Botta, C. Ionescu, and G. Chen, "ECOD: Unsupervised outlier detection using empirical cumulative distribution functions," *TKDE*, 2022.
[6] Y. Almardeny, N. Boujnah, and F. Cleary, "A novel outlier detection method for multivariate data," *TKDE*, 2020.
[7] H. Späth, "The minisum location problem for the Jaccard metric," *Operations-Research-Spektrum*, 1981.
[8] G. A. Watson, "An algorithm for the single facility location problem using the Jaccard metric," *SIAM J. Sci. Stat. Comput.*, 1983.
[9] F. Chierichetti, R. Kumar, S. Pandey, and S. Vassilvitskii, "Finding the Jaccard median," in *SODA*, 2010.
[10] C. Donnat and S. Holmes, "Tracking network dynamics: A survey using graph distances," *AOAS*, 2018.
[11] T. A. Sørensen, "A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons," *Biol. Skar.*, 1948.
[12] R. Bellman, "On the approximation of curves by line segments using dynamic programming," *Commun. ACM*, 1961.
[13] S. E. Amiri, L. Chen, and B. A. Prakash, "Automatic segmentation of dynamic network sequences with node labels," *TKDE*, 2017.
[14] H.-L. Li, "A global approach for general 0-1 fractional programming," *EJOR*, 1994.
[15] Y. Yang, D. Xu, F. Nie, S. Yan, and Y. Zhuang, "Image clustering using local discriminant models and global integration," *TIP*, 2010.
[16] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.
[17] A. Tsitsulin, D. Mottin, P. Karras, A. Bronstein, and E. Müller, "NetLSD: hearing the shape of a graph," in *KDD*, 2018.