Graph Signatures: Identification and Optimization

Balabhaskar Balasundaram*, Juan S. Borrero, and Hao Pan

School of Industrial Engineering and Management, Oklahoma State University, USA

Abstract

We introduce a new graph-theoretic paradigm called a *graph signature* that describes persistent patterns in a sequence of graphs. This framework is motivated by the need to detect subgraphs of significance in temporal networks, e.g., social and biological networks that evolve over time. Because the subgraphs of interest may not all "look alike" in the snapshots of the temporal network, the framework deems a subgraph to be *persistent* if it satisfies one of several preselected properties in each snapshot of a consecutive subsequence. The persistency requirement is parameterized by the length of this subsequence. This discrete mathematical framework can be viewed more broadly as a way to generalize classical graph properties and invariants associated with a single graph to a sequence of graphs.

In this introductory article, we formulate the *graph signature identification problem* as a mixed-integer program and propose an algorithmic framework based on dynamic programming. This methodology is applicable to any collection of mixed-integer representable graph properties. We also demonstrate how this framework can be tailored to exploit property-specific decomposition and scale reduction techniques through three different computational case-studies. Our experiments show that the dynamic programming algorithm solves this problem across most instances in our test bed to optimality. Moreover, for the instances in our test bed, the optimal signature sizes are comparable to those of their static counterparts, suggesting that our new framework can identify subgraphs of significance in complex dynamic networks.

1 Background

Cluster detection in social and biological networks involves modeling clusters of interest using cliques or their graph-theoretic generalizations. Clusters in social networks may represent cohesive social subgroups and could be identified for use in recommender systems, marketing campaigns, community detection, and influence maximization (Alhajj & Rokne, 2018). In biological networks like protein-protein interaction networks, gene co-expression networks, and metabolic networks, detection of clusters or other types of network motifs is commonly used to identify functional modules that could represent protein complexes, transcriptional modules, or signaling pathways (Junker & Schreiber, 2008). Detection methods seek to find a subset of vertices that satisfy the requirements of the selected graph model, typically one that optimizes some direct or indirect measure of fitness like cardinality, weight, or other attributes.

Broadly, optimization approaches to mining graph models of data predominantly share two common characteristics.

*Corresponding author: baski@okstate.edu

(a) They identify cohesive subgraphs, critical nodes, most central actors, or other graph structures of interest in a *single* graph snapshot (Balasundaram et al., 2011; Walteros & Pardalos, 2012; Veremyev et al., 2014; Vogiatzis et al., 2015).

(b) They model structures of interest using a *single* graph-theoretic concept like clique, domination, centrality, connectivity, etc., (Seidman & Foster, 1978; Kelleher & Cozzens, 1988; Borgatti et al., 1990; Freeman, 1992).

The motivating applications in social and biological network analysis as well as in other areas often yield time-series data, which in turn corresponds to a sequence of graphs over time in which the vertices and the edges change (Hu et al., 2005; Li et al., 2011; Hellmann & Staudigl, 2014; Paranjape et al., 2017). Independently analyzing each snapshot graph ignores dynamic characteristics like the *persistence of a pattern in time*. It is also necessary to accommodate the possibility that the vertex subsets of interest could induce *subgraphs that appear very different over time*, resembling one of many different graph-theoretic structures like a clique, a dense subgraph, or a sparse low-diameter subgraph as the interactions (edges) become active or dormant with time.

Works that recognize these challenges and at least partially address them have appeared in literature. A popular approach, with a large body of related work, involves finding *frequent subgraphs* in multiple graphs (Jiang et al., 2013), similar to the use of frequently occuring network motifs in social and biological network analysis (Paranjape et al., 2017). The sequence of graph snapshots is referred to as *relational graphs* or *transactional graphs* in this literature. In this setting, one seeks to find a subgraph, typically required to be dense or connected, that is *recurrent*, i.e., present in at least a minimum number of snapshots; or alternately, find a subgraph of an auxiliary graph (that summarizes the information) whose edges are present in at least a minimum number of snapshots (cf. Kuramochi & Karypis (2005); Hu et al. (2005); Yan et al. (2005)). However, counting occurrences alone may not be a satisfactory surrogate for persistency, as it is insensitive to the ordering of the snapshots.

More recently, some authors have taken the following approach to find Δ-*cliques* in temporal networks (Viard et al., 2016; Himmel et al., 2017). Consider a subset of vertices C and an interval of consecutive graph snapshots $\mathcal{I}$ that share a common vertex set. The subset C is called a Δ-clique in $\mathcal{I}$ if every pair of vertices in C are adjacent in at least one snapshot out of every Δ consecutive snapshots in $\mathcal{I}$. In other words, any pair of vertices in a Δ-clique over an interval of snapshots cannot be non-adjacent for more than Δ consecutive snapshots. The general approach here is to impose a minimum periodicity in the "atomic" property (required of each vertex, edge, vertex-pair etc.) that the subgraph must satisfy in the static counterpart. Although when $\Delta > 0$, it is possible that C may not be a clique in any graph in the subsequence $\mathcal{I}$, when $\Delta = 0$, C is a clique in every snapshot in $\mathcal{I}$. Hence, this approach is related to the notion of a persistent clique signature we consider in Section 5. Viard et al. (2016) propose an algorithm to enumerate all maximal Δ-cliques, where maximality is by inclusion with respect to both C and $\mathcal{I}$. Viard et al. (2018) generalize their Δ-clique model for instantaneous link streams to enumerating maximal cliques that are persistent over a duration, an approach even more closely related to the persistency requirements of graph signatures we introduce in this article. However, we also require that $\mathcal{I}$ be of a user-specified minimum length.

Although our terminology was inspired by a different source (Defense Advanced Research Projects Agency, 2011), it is interesting to note that Viard et al. (2016) also remark that "In real-world situations ... Δ-cliques are *signatures* [emphasis added] of meetings, discussions, or distributed applications for instance" (p. 245). This is also the reason for many traditional applications of graph-based data mining increasingly needing network models and algorithms that explicitly capture the temporal aspect (Hu et al., 2005; Li et al., 2011; Hellmann & Staudigl, 2014). It is therefore

unsurprising to see a spurt in recent literature on this topic. Bentert et al. (2019) have recently extended the Δ-clique approach to Δ-k-plexes, based on a clique relaxation model for cohesive social subgroups in social network analysis (Seidman & Foster, 1978). The recent work of Latapy et al. (2018) to generalize many basic graph theory concepts to deal with temporal networks, or what they refer to as *stream graphs*, is also of interest in this context.

Another branch of recent literature closely aligned with the modeling approach we introduce in Section 2 introduces the following ideas. Jethava & Beerenwinkel (2015) considered the problem of finding a *densest common subgraph* (DCS) in a sequence of graphs with a common vertex set. That is, a subset of vertices that maximizes the minimum average degree[1] over the subgraphs it induces in every graph in the sequence. Semertzidis et al. (2019) extend this idea in two directions: first, they introduce four variants of the DCS problem by considering average or minimum degree of the subgraph induced by a subset of vertices, aggregated into an objective function by considering the minimum or average over all the snapshots in the sequence; second, they introduce a relaxation that requires the subset of vertices to optimize one of the measures of density over a subsequence of k snapshots. It is important to note that the k snapshots need not be consecutive in their framework. This characteristic, inspired by thinking not unlike that underlying the frequent subgraph models, also differentiates their approach from ours. The four variants introduced by Semertzidis et al. (2019) were further investigated by Charikar et al. (2018), with both studies offering greedy algorithms, approximation and complexity results for these problems.

In the next section, we introduce a new general-purpose combinatorial optimization framework for identifying structures of interest in temporal graphs (digraphs or multigraphs). Our framework is distinguishable from prevailing approaches in how the need for *persistency* and model *adjustability* is captured when analyzing temporal networks. In particular, we propose to model the signature problem using mixed-integer programming, and develop generic formulations and a dynamic programming-based algorithm to solve the resulting optimization problems.

The remainder of this paper is organized as follows. Section 2 formally describes the problem of interest and highlights some of its salient characteristics. Integer programming formulations and a dynamic programming approach for the generic problem are introduced in Sections 3 and 4, respectively. Specific model case-studies are undertaken in Sections 5, 6, and 7, presenting problem-specific computational enhancements for solving the associated optimization problems. Section 8 presents the results of our computational experiments to assess and compare the performance of the solution approaches introduced. The article is concluded with some remarks on future extensions in Section 9.

2 Persistent Graph Signatures

Given a positive integer a, we use the notation $[a] \coloneqq \{1, \ldots, a\}$ to compactly denote index sets. Suppose we are given a non-empty finite collection of graph properties applicable to vertex subsets, denoted by $\mathcal{P}$, and a sequence of graphs denoted by $\mathcal{G} \coloneqq (G^t, t \in [T])$. We often think of $[T]$ as a discretized time-horizon of interest, although this interpretation is not strictly required. Each graph $G^t = (V^t, E^t)$ is a subgraph of a universal graph $G^0 = (V^0, E^0)$ of order n with $V^0 \coloneqq [n]$. Note that we have not made an explicit assumption as to whether G^t is a "simple" graph, free from loops and parallel edges, a digraph, or a (directed) multigraph. Naturally in the last case we will also need a mapping from edges to end-points to describe parallel edges.

[1]Note that the authors define as *density*, the number edges in the induced subgraph divided by the order, i.e., half the average degree of the induced subgraph. Hence, the density metric they optimize is proportional to average degree of the induced subgraph.

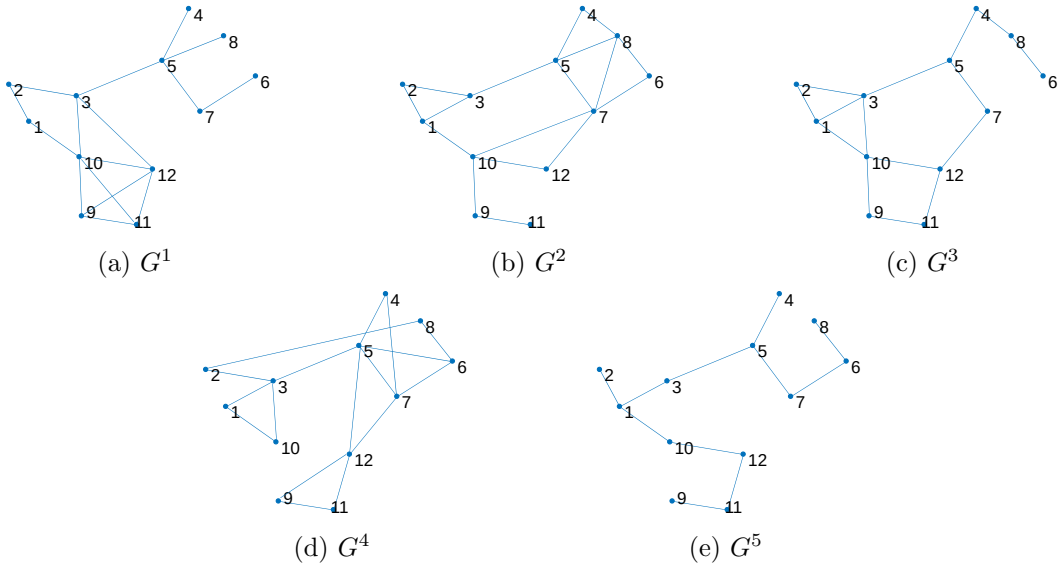


Figure 1: A sequence of $T = 5$ snapshot graphs. The size of the maximum 3-persistent 2-club is four, corresponding to $\{1, 2, 3, 10\}$.

Definition 1. Consider a graph sequence $\mathcal{G}$, a collection of graph properties $\mathcal{P}$, a positive integer τ, and a subset of vertices $S \subseteq V^0$. We say that S is a *τ-persistent $\mathcal{P}$-signature* in $\mathcal{G}$ if there exists a subsequence $(G^t, \ldots, G^{t+\tau-1})$ of $\mathcal{G}$ such that, in every graph in this subsequence, subset S satisfies at least one property $\pi \in \mathcal{P}$. Succinctly, we state that S satisfies condition $\Sigma(\mathcal{G}, \mathcal{P}, \tau)$.

The *graph signature identification problem* (GSIP) is to find a τ-persistent $\mathcal{P}$-signature $S^* \subseteq V^0$ in $\mathcal{G}$ that maximizes an objective function $w : 2^{V^0} \longrightarrow \mathbb{R}^+$. In other words, the set S^* solves the following optimization problem (1) and we say S^* is an optimal graph signature:

$$\textbf{GSIP:} \qquad \sigma(\mathcal{G}, \mathcal{P}, \tau) \coloneqq \max_{S \subseteq V^0} \{w(S) \mid S \text{ satisfies } \Sigma(\mathcal{G}, \mathcal{P}, \tau)\}. \tag{1}$$

In order to illustrate an optimal graph signature, let us consider the example where $\mathcal{P} = \{2\text{-Club}\}$. A 2-club is a subset of vertices in which non-adjacent pairs of vertices are required to have a common neighbor in the subset. This is a popular model for cohesive subgroups in social network analysis as it represents a group of people in which every pair of strangers have a mutual friend. Note that every clique is also a 2-club. Suppose that the objective function is $w(S) = |S|$, the cardinality of the subset of vertices, and let $\tau = 3$. In the graph sequence $\mathcal{G} = (G^1, G^2, \ldots, G^5)$ in Figure 1, the optimal graph signature is $S^* = \{1, 2, 3, 10\}$ with an optimal value of $\sigma(\mathcal{G}, \mathcal{P}, \tau) = 4$. Observe that in this example there are larger 2-clubs, such as $\{1, 3, 9, 10, 11, 12\}$ in snapshot G^1 and $\{4, 5, 6, 7, 8\}$ in snapshot G^2. These subsets, however, do not exhibit persistency for more than one period.

The first noteworthy modeling characteristic of a graph signature is the assumption that multiple properties of interest may exist, as captured by the set $\mathcal{P}$. For instance, a subset of vertices may induce a clique in most of the graphs in a subsequence of length τ, but it may be missing a few edges in some of them. However, when it fails to satisfy the definition of a clique in some of the snapshots in the subsequence, it may still satisfy the definition of a k-plex for some positive integer k, which is a hereditary, degree-based clique relaxation that allows for up to $k - 1$ non-neighbors in the subgroup for every vertex (Seidman & Foster, 1978; Balasundaram et al., 2011). Such a signature could be detected by defining $\mathcal{P}$ as a 3-plex for instance, because every clique is a 2-plex

and every 2-plex is a 3-plex. Parameterized graph properties like clique relaxations (Pattillo et al., 2013) are natural candidates for $\mathcal{P}$. Alternatively, $\mathcal{P}$ could include incomparable properties such as minimum edge density of 70% and maximum diameter of two, i.e., $\mathcal{P} = \{0.7\text{-QUASICLIQUE}, 2\text{-CLUB}\}$. This type of $\mathcal{P}$ may be necessary, for instance, in covert social networks in which the ties between the majority of actors may become dormant when all communication happens via a central actor. Then, the density of the cohesive social subgroup might dramatically drop, but the group may continue to induce a low-diameter subgraph.

The second modeling characteristic that is important is the notion of persistency. Although we permit the vertex subset S to satisfy one of many properties and not necessarily the same property over time, it must do so continuously over a consecutive subsequence of some minimum length, in order to be deemed interesting, i.e., a signature that is persistent over time. By parameterizing this minimum length using τ, we can relax or restrict the persistency requirement as demanded by the application.

These key ideas make the overall framework of graph signatures potentially more useful than approaches seeking a cluster that satisfies a single graph property in a single graph snapshot. Moreover, our framework complements and extends many of the prevailing approaches for analyzing temporal networks reviewed in Section 1. Mathematically, this framework is flexible and adjustable, and potentially has greater applicability in different domains including social and biological network analysis.

3 GSIP Formulations

We introduce additional notations and assumptions before formulating the generic GSIP as a mathematical program. Analogous to problem (1), for any subsequence $\widetilde{\mathcal{G}}$ of $\mathcal{G}$, we define $\sigma(\widetilde{\mathcal{G}}, \mathcal{P}, \tau)$ by only considering those graphs G^t of $\mathcal{G}$ that appear in the subsequence $\widetilde{\mathcal{G}}$. If $\mathcal{P} = \{\pi\}$, a singleton, we use simpler notations to denote the invariants, e.g., $\sigma(\mathcal{G}, \pi, \tau)$ instead of $\sigma(\mathcal{G}, \{\pi\}, \tau)$. We say a vector $x \in \{0,1\}^n$ is an incidence vector of a subset $S \subseteq V^0$, if $S = \{u \in V^0 \mid x_u = 1\}$. We denote this bijection as $x \leftrightarrow S$. In this initial study of the problem, we make the following simplifying assumptions:

A-1. For every $\pi \in \mathcal{P}$ and $t \in [T]$, we assume that S satisfies property π in G^t if and only if there exists a matrix $\boldsymbol{A}^t_\pi$ and a vector $\boldsymbol{b}^t_\pi$ such that $\boldsymbol{A}^t_\pi x \leq \boldsymbol{b}^t_\pi$ for $x \leftrightarrow S$. In other words, there exists a mixed integer linear programming (MILP) formulation for each property $\pi \in \mathcal{P}$, projected onto the x-variable space if necessary, so that the collection of feasible incidence vectors denoted by Q^t_π satisfies the following equation:

$$Q^t_\pi = \left\{x \in \{0,1\}^n \mid \boldsymbol{A}^t_\pi x \leq \boldsymbol{b}^t_\pi\right\}.$$

A-2. There exists a diagonal matrix of "big-M" values denoted by $\boldsymbol{M}^t_\pi$ of appropriate dimension such that $\boldsymbol{A}^t_\pi x \leq \boldsymbol{b}^t_\pi + \boldsymbol{M}^t_\pi \mathbf{1}$ holds for every $x \in \{0,1\}^n$, where $\mathbf{1}$ is a vector of ones of appropriate dimension.

A-3. The objective we wish to maximize is linear, i.e. $w(S) = \boldsymbol{w}^\top x$ for $x \leftrightarrow S$.

In addition to the incidence vector x that serves as a vector of decision variables, we also introduce the binary indicator variables z_t for each $t \in [T]$ and y^t_π for each $t \in [T]$ and $\pi \in \mathcal{P}$. Under the foregoing assumptions, the generic GSIP can be formulated as the following nonlinear optimization problem in binary variables.

$$
\begin{aligned}
\textbf{GSIP-F1:}\quad \sigma(\mathcal{G},\mathcal{P},\tau) &= \max \boldsymbol{w}^\top x && && \text{(2a)}\\
\text{s.t.}\quad \boldsymbol{A}^t_\pi x &\le \boldsymbol{b}^t_\pi + \boldsymbol{M}^t_\pi(\mathbf{1} - y^t_\pi \mathbf{1}) && \forall t \in [T], \pi \in \mathcal{P} && \text{(2b)}\\
\sum_{t\in[T-\tau+1]} \prod_{j=t}^{t+\tau-1} z_j &\ge 1 && && \text{(2c)}\\
z_t &\le \sum_{\pi\in\mathcal{P}} y^t_\pi && \forall t \in [T] && \text{(2d)}\\
x_u &\in \{0,1\} && \forall u \in V^0 && \text{(2e)}\\
y^t_\pi &\in \{0,1\} && \forall t \in [T], \pi \in \mathcal{P} && \text{(2f)}\\
z_t &\in \{0,1\} && \forall t \in [T] && \text{(2g)}
\end{aligned}
$$

Consider a feasible solution (x, y, z) to formulation (2), which we henceforth refer to as **GSIP-F1**. Constraint (2b) implies that if $y^t_\pi = 1$, then $x \in Q^t_\pi$. Constraint (2c) requires that at least τ consecutive z_t variables must be at one in any feasible solution. Constraint (2d) enforces that if $z_t = 1$, then at least one of the y^t_π variables for $\pi \in \mathcal{P}$ must be one. This along with constraint (2b) ensures that if $z_t = 1$, then $S \leftrightarrow x$ satisfies some property $\pi \in \mathcal{P}$ in G^t. Therefore, $x \in \{0,1\}^n$ is a feasible solution of formulation (2) if and only if $S \leftrightarrow x$ satisfies condition $\Sigma(\mathcal{G},\mathcal{P},\tau)$.

If $\mathcal{P} = \{\pi\}$, a singleton, the definition and the values of variables y^t_π and z_t coincide. In other words, $y^t_\pi = z_t$ for every $t \in [T]$ for a feasible solution (x, y, z). In this case, we can replace y^t_π with z_t in formulation (2), drop the redundant constraints, and obtain the simplified formulation (3), which we will call **GSIP-F2**. Note that we drop the subscript π to make it apparent that only one property is involved.

$$
\begin{aligned}
\textbf{GSIP-F2:}\quad \sigma(\mathcal{G},\pi,\tau) &= \max \boldsymbol{w}^\top x && && \text{(3a)}\\
\text{s.t.}\quad \boldsymbol{A}^t x &\le \boldsymbol{b}^t + \boldsymbol{M}^t(\mathbf{1} - z_t \mathbf{1}) && \forall t \in [T] && \text{(3b)}\\
&\text{(2c)}, \text{(2e)}, \text{(2g)}
\end{aligned}
$$

GSIP-F1 and **GSIP-F2** may be infeasible if there is no τ-persistent graph signature in $\mathcal{G}$. We follow the convention that $\sigma(\mathcal{G},\mathcal{P},\tau)$ and $\sigma(\mathcal{G},\pi,\tau)$ are $-\infty$ whenever the formulations are infeasible. Generally, if at least one property in $\mathcal{P}$ is satisfied by the empty set, by a single vertex (e.g., clique is in $\mathcal{P}$), or the universal vertex set V^0 (e.g., domination or vertex cover is in $\mathcal{P}$), the formulations should always have a feasible solution.

4 GSIP and Dynamic Programming

Smaller instances of the graph signature problem can be solved using an off-the-shelf MILP solver using a linearization of **GSIP-F1** or **GSIP-F2**, as applicable. However, the number of variables, constraints, and non-zeros of such monolithic formulations increase linearly as the number of time periods T increases, rendering them impractical for large values of T, even on moderately sized graphs. Alternatively, we could use a dynamic programming approach, which involves solving $T - \tau + 1$ smaller subproblems in a moving-window like fashion by only considering τ consecutive time-periods for each subproblem.

Specifically, consider $t = \tau, \tau + 1, \ldots, T$, and define f^t as the value of the maximum weight τ-persistent signature until time t, i.e., $f^t \coloneqq \sigma(\mathcal{G}^t,\mathcal{P},\tau)$, where $\mathcal{G}^t \coloneqq (G^r, r \in [t])$. Then, it is

readily seen that,

$$f^t = \max\left\{f^{t-1}, \sigma(\mathcal{G}^{t,\tau}, \mathcal{P}, \tau)\right\} \qquad \text{for } t = \tau+1, \ldots, T, \tag{4}$$

where $\mathcal{G}^{t,\tau}$ is the sequence of graphs defined by $\mathcal{G}^{t,\tau} \coloneqq (G^r, r \in \{t-\tau+1, \ldots, t\})$. As a consequence, we can compute $f^T = \sigma(\mathcal{G}, \mathcal{P}, \tau)$ by first computing f^τ, noting that $f^\tau = \sigma(\mathcal{G}^\tau, \mathcal{P}, \tau)$, and then using the recursion (4) to compute f^r from period $r = \tau+1$ until period $r = T$. Hereafter, we refer to this dynamic programming recursion as the *Moving-Window (MW) method*.

In the MW method, $\sigma(\mathcal{G}, \mathcal{P}, \tau)$ is computed by solving $T-\tau+1$ different optimization problems. However, these problems do not need the z_t variables or the nonlinear constraints (2c) that **GSIP-F1** and **GSIP-F2** use. Indeed, formulations (5) and (6) presented next, which we refer to as **GSIP-F1-MW** and **GSIP-F2-MW** are valid for $\sigma(\mathcal{G}^{t,\tau}, \mathcal{P}, \tau)$ and $\sigma(\mathcal{G}^{t,\tau}, \pi, \tau)$, respectively. In addition, **GSIP-F2-MW** does not require the use of "big-M" coefficients in contrast to formulation **GSIP-F2**.

GSIP-F1-MW:

$$\sigma(\mathcal{G}^{t,\tau}, \mathcal{P}, \tau) = \max\ \boldsymbol{w}^\top x \tag{5a}$$

$$\text{s.t. } \boldsymbol{A}^r_\pi x \le \boldsymbol{b}^r_\pi + \boldsymbol{M}^r_\pi(\mathbf{1} - y^r_\pi \mathbf{1}) \qquad \forall r = t-\tau+1, \ldots, t,\ \pi \in \mathcal{P} \tag{5b}$$

$$\sum_{\pi \in \mathcal{P}} y^r_\pi \ge 1 \qquad \forall\ r = t-\tau+1, \ldots, t \tag{5c}$$

$$x_u \in \{0,1\} \qquad \forall u \in V^0 \tag{5d}$$

$$y^r_\pi \in \{0,1\} \qquad \forall \pi \in \mathcal{P}; r = t-\tau+1, \ldots, t \tag{5e}$$

GSIP-F2-MW:

$$\sigma(\mathcal{G}^{t,\tau}, \pi, \tau) = \max\ \boldsymbol{w}^\top x \tag{6a}$$

$$\text{s.t. } \boldsymbol{A}^r x \le \boldsymbol{b}^r \qquad \forall r = t-\tau+1, \ldots, t \tag{6b}$$

$$x_u \in \{0,1\} \qquad \forall u \in V^0 \tag{6c}$$

Formulation (6), or more precisely the underlying optimization problem when $\tau = 2$ is known as the *cross-graph mining problem* in the graph-based data mining literature (Jiang & Pei, 2009; Pei et al., 2005). **GSIP-F2-MW** and **GSIP-F1-MW** can therefore be viewed as its generalization over multiple graphs and multiple graph properties.

Given the popularity of cliques as the gold-standard for modeling "ideal" clusters in graph-based data mining (Cook & Holder, 2006) as well as its position as the benchmark for modeling cohesive subgroups in social networks (Wasserman & Faust, 1994), we consider clique signatures a canonical graph signature identification problem. This is the first special case we consider in Section 5. Then in Section 6 we consider the 2-club model briefly introduced in Section 2, a distance-based clique relaxation used to model cohesive social subgroups. Unlike cliques, the 2-club property is not preserved under vertex deletions in general (i.e., it is not hereditary on induced subgraphs) and requires decomposition ideas specifically tailored to this problem. In contrast to cliques and 2-clubs, finding a maximum k-core in a graph is a polynomial-time solvable problem (Seidman, 1983). In Section 7, we consider k-core signatures and introduce a combinatorial polynomial-time algorithm for the problem.

In the remainder of this article, we make the following additional assumptions.

A-4. The vertex set is fixed across all time periods, thus $V^t = V^0$ for all $t \in [T]$; we lose no generality by doing so for the properties we consider.

A-5. For each period $t \in [T]$, the graph $G^t = (V^t, E^t)$ is undirected, unweighted, and without loops or parallel edges. Hence,

$$E^t \subseteq \binom{V^t}{2} \coloneqq \left\{ \{u, v\} \;\middle|\; u, v \in V^t, u \neq v \right\}.$$

A-6. We focus on the case of singleton $\mathcal{P}$.

A-7. We focus on maximum *cardinality* τ-persistent signatures. That is, we set the objective vector $\boldsymbol{w} = \mathbf{1}$.

5 Clique Signatures

This section discusses how persistent clique signatures can be found. We begin with a well-known formulation of the maximum clique problem, which forms the basis for the formulations used in the MW method. Then, we show how the clique signature problem is equivalent to the maximum clique problem on an auxiliary graph and how the MW method can be enhanced by exploiting this fact.

Incidence vectors of cliques in G^t, for a given time period $t \in [T]$, can be formulated using the well-known complement edge constraints (7). Denoting the complement graph by $\overline{G}^t = (V^t, \overline{E}^t)$, where $\overline{E}^t \coloneqq \binom{V^t}{2} \setminus E^t$, we can write these constraints as,

$$x_i + x_j \leq 1 \qquad \forall \{i, j\} \in \overline{E}^t. \tag{7}$$

Constraints (7) can be used in **GSIP-F2** and **GSIP-F2-MW** by noting that $\boldsymbol{A}^t$ is precisely the edge-vertex incidence matrix of the complement graph $\overline{G}^t$ and $\boldsymbol{b}^t$ is a vector of ones of appropriate size.

Applying the MW method (4) to find a τ-persistent clique signature requires repeatedly finding a maximum clique over τ consecutive graphs in $\mathcal{G}$. For a given time period $t \in \{\tau, \ldots, T\}$, this can be accomplished by solving formulation (6), which in this case reduces to the following.

$$\sigma(\mathcal{G}^{t,\tau}, \textsc{Clique}, \tau) = \max \; \mathbf{1}^\top x \tag{8a}$$

$$\text{s.t. } x_i + x_j \leq 1 \quad \forall \{i, j\} \in \bigcup_{r=t-\tau+1}^{t} \overline{E}^r \tag{8b}$$

$$x_i \in \{0, 1\} \quad \forall i \in V^0 \tag{8c}$$

Observe that formulation (8) is precisely the complement edge formulation of the maximum clique problem of the *intersection graph* $H^{t,\tau}$, where,

$$H^{t,\tau} = \left(V^0, \bigcap_{r=t-\tau+1}^{t} E^r \right).$$

In other words, finding a maximum clique signature on $\mathcal{G}^{t,\tau}$ reduces to finding the maximum clique on the corresponding intersection graph. By using the MW method (4), the maximum clique signature of $\mathcal{G}$ can be computed by solving $T - \tau + 1$ maximum clique problems on the (potentially) sparser graphs $H^{t,\tau}$ for each $t = \tau, \ldots, T$. We summarize this relationship in Proposition 1.

Proposition 1. Consider a sequence of graphs $\mathcal{G} = (G^t, t \in [T])$ and a positive integer τ. A subset of vertices S is a τ-persistent clique signature in $\mathcal{G}$ if and only if there exists a $t \in \{\tau, \ldots, T\}$ such that S is a clique in the intersection graph $H^{t,\tau}$.

Proposition 1 allows us to exploit well known computational methods available for the maximum clique problem (Verma et al., 2015; Rebennack et al., 2011; Abello et al., 1999). Specifically, we dynamically apply scale reduction techniques such as core peeling and community peeling to the intersection graph to enable a general-purpose branch-and-cut based MILP solver to reduce the time it takes to find a maximum clique.

Core peeling for clique signatures works as follows (Abello et al., 1999). If at time $t \in \{\tau + 1, \ldots, T\}$ we know that $f^{t-1} = \ell$ (see Equation (4)), then in $H^{t,\tau}$ we are only interested in cliques of size $\ell + 1$ or more. Hence, we can recursively delete any vertex that has degree less than ℓ. In other words, we can find the ℓ-core of $H^{t,\tau}$, the unique maximal subgraph of minimum degree ℓ (Seidman, 1983), and solve the maximum clique problem on the ℓ-core. If the ℓ-core is a null graph, then $H^{t,\tau}$ does not have a clique of size at least $\ell + 1$ and we can set $f^t = \ell$. However, if the ℓ-core is not null, we apply *community peeling*, as described next.

We know that any pair of vertices in a clique of size $\ell + 1$ or more should have at least $\ell - 1$ common neighbors. Hence, we can recursively delete every edge in the ℓ-core of $H^{t,\tau}$ whose endpoints do not satisfy this condition. The resulting graph in which the endpoints of every edge have at least $\ell - 1$ common neighbors is called an $(\ell - 1)$-community (Verma et al., 2015). If the resulting graph is empty, then the clique of size ℓ is maximum, that is, $f^t = \ell$; otherwise, we solve the maximum clique problem on the resulting $(\ell - 1)$-community graph. We apply core peeling before community peeling based on past computational studies that demonstrate it is more effective to core-peel a graph before applying community peeling (Verma et al., 2015).

Besides peeling, an extended formulation can be used to facilitate the MILP solver to recognize and exploit the fact that the clique must be contained within a connected component. We can extend the complement edge formulation (8) by introducing new binary variables that indicate the component in which the clique is present. This componentwise formulation typically results in fewer non-zero coefficients when compared with a formulation that ignores the fact that the graph is disconnected. In formulation (9), we use the short form $\forall(i, j, C)$ to refer to every connected component C in the intersection graph H of interest and every distinct pair of non-adjacent vertices in component C. Let $\mathcal{C}$ denote the collection of vertex sets of each connected component of H, then the formulation is as follows:

$$\omega(H) = \max \mathbf{1}^\top x \tag{9a}$$
$$\text{s.t. } x_i + x_j \leq 1 \quad \forall(i, j, C) \tag{9b}$$
$$\sum_{C \in \mathcal{C}} \xi_C \leq 1 \tag{9c}$$
$$x_i \leq \xi_C \quad \forall i \in C \text{ and } C \in \mathcal{C} \tag{9d}$$
$$x_i \in \{0, 1\} \quad \forall i \in V(H) \tag{9e}$$
$$\xi_C \in \{0, 1\} \quad \forall C \in \mathcal{C}. \tag{9f}$$

Algorithm 1 describes our overall approach that implements the recursion (4) for finding persistent clique signatures. In step 1 of Algorithm 1, a maximal clique is found by recursively selecting and adding a vertex of maximum degree in the graph induced by candidate vertices, i.e., those that are adjacent to all the vertices in the current clique (Bomze et al., 1999). The initial heuristic

solution is used in the scale reduction techniques for the first window, and the best solution is updated and used in subsequent windows.

Algorithm 1: (MW-CLQ) Finding a maximum clique signature using the moving window method.

Input: $\mathcal{G} = (G^t, t \in [T])$, $\tau \geq 1$, $\ell = 0$.
Output: A maximum cardinality clique signature S.

$S \leftarrow$ a maximal clique in the intersection graph $H^{\tau,\tau}$
for $t = \tau, \ldots, T$ **do**
  $\ell \leftarrow |S|$
  $H^{t,\tau} \leftarrow \textsc{CorePeel}(H^{t,\tau}, \ell)$
  $H^{t,\tau} \leftarrow \textsc{CommunityPeel}(H^{t,\tau}, \ell)$
  $S' \leftarrow$ solve for a maximum clique in $H^{t,\tau}$ using formulation (9)
  $S \leftarrow \arg\max\{|S|, |S'|\}$
return S

6 2-Club Signatures

Although cliques describe an ideal notion of clusters or cohesive social subgroups, in graphs built from error-prone or incomplete data a cluster of significance might not resemble a clique in every snapshot of the graph sequence. Therefore, even though these clusters might be 'almost' cliques for many consecutive periods, they cannot be considered τ-persistent cliques. In order to capture signatures of this type, we can include one or more parameterized graph-theoretic clique relaxations in $\mathcal{P}$ (Balasundaram & Pajouh, 2013; Pattillo et al., 2013). In this introductory article we restrict our attention to a well-known distance-based clique relaxation, the 2-club model (Shahinpour & Butenko, 2013).

Definition 2 (Alba (1973); Mokken (1979); cf. Balasundaram et al. (2005))**.** Given a graph $G = (V, E)$, a subset of vertices S is called a 2-club in G if every distinct pair of vertices are either adjacent, or have a common neighbor inside S.

Clearly, the 2-club model generalizes cliques by allowing non-adjacent pairs of vertices in the cluster as long as they have a common neighbor in the cluster. As noted before, this model is well suited for applications in which cohesive social subgroups can contain strangers as long as they have a mutual friend, the so-called friend-of-a-friend cluster. It has found use in a variety of domains as a result (Miao & Berleant, 2004; Terveen et al., 1999).

As a model for a τ-persistent cohesive subgroup signature, the 2-club is more flexible because it not only includes cliques but sparser subgroups as well (e.g., a vertex and all its neighbors meet the requirement). In this article we only consider 2-club signatures, noting that both cliques and 2-clubs are special cases of a more general distance-based clique relaxation called an s-club (Shahinpour & Butenko, 2013). Our approach to finding 2-club signatures can be extended to find τ-persistent s-club signatures using prevailing approaches for solving the maximum s-club problem (Salemi & Buchanan, 2020; Moradi & Balasundaram, 2018).

In order to provide an MILP formulation for 2-clubs, we use the "common neighbor" formulation (Bourjolly et al., 2002). Let $N_{G^t}(i)$ denote the set of neighbors of a vertex $i \in V^t$, i.e., $N_{G^t}(i) := \left\{ j \in V^t \;\middle|\; \{i, j\} \in E^t \right\}$. Then, a set $S \leftrightarrow x$ is a 2-club in G^t if it satisfies the common

neighbor constraints (10):

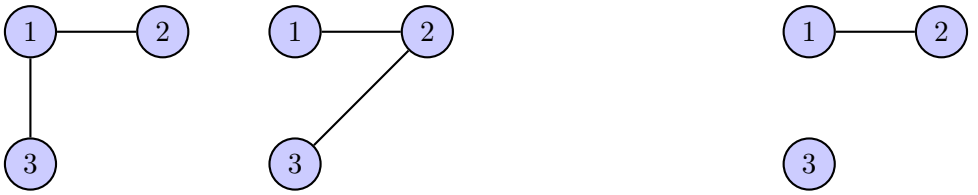
$$x_i + x_j - \sum_{k \in N_{G^t}(i) \cap N_{G^t}(j)} x_k \leq 1 \qquad \forall \{i,j\} \in \overline{E}^t. \tag{10}$$

It is worth noting that a naive extension of the intersection graph approach used with clique signatures is not valid for 2-club signatures. Indeed, a set of vertices can be a 2-club signature in $\mathcal{G}^{t,\tau}$ without being one in the intersection graph $H^{t,\tau}$ (see Figure 2). Consequently, we would restrict ourselves (incorrectly) if we only seek 2-clubs in $H^{t,\tau}$. Recall that it was the equivalence between the maximum τ-persistent clique signature problem and the maximum clique problem on the intersection graph $H^{t,\tau}$ that permitted us to exploit established computational ideas that are known to be effective for solving the maximum clique problem. Although such an equivalence is not readily available, we can still take advantage of preprocessing and decomposition techniques by considering a relaxation of the τ-persistent maximum 2-club signature problem.

Consider the graph in Figure 3. Clearly, the filled vertices do not form a 2-club as the induced subgraph has two vertices that do not share a common filled neighbor, highlighting that the 2-club property is not hereditary on vertex induced subgraphs. The filled vertices form what is called a distance-2-clique, or in short, a *2-clique*. In order to formalize this notion, let $\text{dist}_G(u,v)$ be the distance between two vertices u and v in G. That is, $\text{dist}_G(u,v)$ is the minimum length of a path (measured in number of edges) connecting them in G.

Definition 3 (Luce (1950); cf. Balasundaram et al. (2005))**.** Given a graph $G = (V,E)$, a subset of vertices S is called a 2-clique in G if every pair of vertices $u, v \in S$ satisfy the condition $\text{dist}_G(u,v) \leq 2$.

Clearly, every 2-club is a 2-clique, but not vice versa. Furthermore, a subset of vertices S is a 2-clique in $G = (V,E)$ if and only if S is a clique in the *square graph* $\widetilde{G} = (V, \widetilde{E})$, where $\widetilde{E} \coloneqq \{\{u,v\} \mid 1 \leq \text{dist}_G(u,v) \leq 2\}$. Because a τ-persistent 2-club signature S must be a 2-club in every graph in $\mathcal{G}^{t,\tau}$ for some $t = \tau, \ldots, T$, we know that S must also be a 2-clique in every



(a) Graphs G^1 and G^2 (b) Intersection graph $H^{2,2}$

Figure 2: This example illustrates that a 2-club signature is not necessarily a 2-club in the intersection graph. The set $\{1,2,3\}$ is a 2-persistent 2-club signature in $\mathcal{G} = (G^1, G^2)$, but it is not a 2-club in the intersection graph.

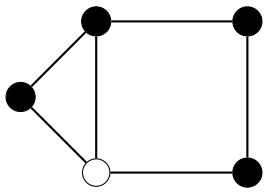


Figure 3: The filled vertices form a distance-2-clique, but not a 2-club.

graph in $\mathcal{G}^{t,\tau}$, or equivalently a clique in the square of every graph in $\mathcal{G}^{t,\tau}$. In other words, we have established the following proposition.

Proposition 2. Consider a sequence of graphs $\mathcal{G} = (G^t, t \in [T])$ and a positive integer τ. Define the *square intersection graph* $J^{t,\tau}$ as,

$$J^{t,\tau} := \left(V^0, \bigcap_{r=t-\tau+1}^{t} E^{r,2} \right), \tag{11}$$

where $E^{r,2} := \{\{u, v\} \mid 1 \leq \text{dist}_{G^r}(u, v) \leq 2\}$. A subset of vertices S is a τ-persistent 2-clique signature in $\mathcal{G}$ if and only if there exists a $t = \tau, \ldots, T$ such that S is a clique in the square intersection graph $J^{t,\tau}$.

Proposition 2 forms the basis for Algorithm 2 for finding a maximum cardinality τ-persistent 2-club signature in $\mathcal{G}^{t,\tau}$. Notice that this algorithm follows the same overall steps as Algorithm 1, with some of the steps requiring further elaboration as discussed next.

Algorithm 2: (MW-2CLB) Finding a maximum cardinality 2-club signature using the moving window method.

Input: $\mathcal{G} = (G^t, t \in [T])$, $\tau \geq 1$, $\ell = 0$.
Output: A maximum cardinality 2-club signature S.

1 $v \leftarrow \arg\max\{|N_{H^{\tau,\tau}}(u)| \mid u \in V^0\}$
2 $S \leftarrow \{v\} \cup N_{H^{\tau,\tau}}(v)$
3 **for** $t = \tau, \ldots, T$ **do**
4 $\quad \ell \leftarrow |S|$
5 $\quad J^{t,\tau} \leftarrow \textsc{CorePeel}(J^{t,\tau}, \ell)$
6 $\quad J^{t,\tau} \leftarrow \textsc{CommunityPeel}(J^{t,\tau}, \ell)$
7 $\quad S' \leftarrow$ solve for a maximum τ-persistent 2-club in $\mathcal{G}^{t,\tau}$ using delayed generation of common neighborhood constraints applied to the maximum clique master relaxation formulation (9) on $J^{t,\tau}$
8 $\quad S \leftarrow \arg\max\{|S|, |S'|\}$
9 **return** S

In step 7 of Algorithm 2 we start by solving the master relaxation—the maximum clique problem on the square intersection graph $J^{t,\tau}$—using a general purpose branch-and-cut solver. (Observe that by Proposition 2, such a maximum clique is a τ-persistent 2-clique on $\mathcal{G}^{t,\tau}$.) Whenever an integral solution is encountered, we must verify if this τ-persistent 2-clique signature in $\mathcal{G}^{t,\tau}$ is indeed a τ-persistent 2-club signature in $\mathcal{G}^{t,\tau}$. If it is, then the corresponding node of the search tree can be pruned by feasibility; if not, we must add constraints that cut off this infeasible integral solution.

The infeasible solution can be removed by noting that it must violate some common neighbor constraint (10) for some graph G^r in $\mathcal{G}^{t,\tau}$. We add the first violated constraint we detect into the "lazy" constraint pool, and the node subproblem is re-solved. This type of delayed constraint generation has been found to be effective in several past computational studies for the maximum s-club problem (Salemi & Buchanan, 2020; Moradi & Balasundaram, 2018; Lu et al., 2018). We therefore use this approach, instead of directly solving a 2-club MILP formulation based on (10).

Finally, the effectiveness of the very first application of core and community peeling steps relies on having a good initial heuristic solution. Because a 2-club that exists in the intersection graph

$H^{\tau,\tau}$ exists in every graph in the sequence $(G^t, t \in [\tau])$, we use a vertex of maximum degree and its neighbors in $H^{\tau,\tau}$ as an initial heuristic solution in step 2 of Algorithm 2.

7 Core Signatures

This section focuses on exploring performance of the MW method on a graph property that can be computed in polynomial time. Specifically, we discuss the persistent k-core signature problem, i.e., we set $\mathcal{P} = \{k\text{-CORE}\}$. As opposed to the maximum clique and 2-club problems, the unique maximal k-core of a graph $G = (V, E)$ can be identified in $O(|E| + |V|)$ time (Matula & Beck, 1983). In this section, we describe an algorithm that runs in linear time in the size of the input and finds the persistent k-core of a graph sequence. This algorithm can then be used in the MW method to find a k-core signature of $\mathcal{G}$.

Definition 4 (cf. Seidman (1983))**.** Given a graph $G = (V, E)$, we call a subset of vertices S a k-core in G if the minimum degree of the induced subgraph $G[S]$ is at least k.

In light of the above definition, we define a persistent k-core of a window $\mathcal{G}^{t,\tau}$ as the subset of vertices S that forms a k-core in every graph in the sequence. We define a persistent k-core as a vertex subset (rather than a subgraph as is the convention) because the subgraphs induced by S are not necessarily identical even though they form a k-core in each graph of the window. Additionally, we do not require maximality in the definition of a k-core because $\mathcal{G}^{t,\tau}$ contains a unique S that is a maximal persistent k-core, as in the single-graph counterpart. Indeed, if S and S' are distinct persistent k-cores in $\mathcal{G}^{t,\tau}$, then $S \cup S'$ is a persistent k-core in $\mathcal{G}^{t,\tau}$ as well. Also note that the unique maximal persistent k-core S in $\mathcal{G}^{t,\tau}$ does not need to be a maximal k-core in each graph in the sequence.

Similarly to the clique and 2-club properties, the k-core property admits an MILP formulation. A set $S \leftrightarrow x$ is a k-core in G^t if and only if it satisfies the following constraints:

$$\sum_{k \in N_{G^t}(i)} x_k \geq k x_i \qquad \forall i \in V. \tag{12}$$

The MILP reformulation could be used to solve the signature problem in a similar manner as cliques and 2-clubs. However, for this setting we describe an exact combinatorial algorithm we refer to as **MW-CORE**, see Algorithm 3, that can be used to compute a persistent k-core of a graph sequence. This algorithm runs in time that is linear in size of the input (i.e., the size of the graphs in the sequence $\mathcal{G}^{t,\tau}$).

MW-CORE works by recursively deleting vertices with degree less than k in any graph in the sequence. The queue Q in Algorithm 3 stores the vertices that are going to be deleted, while the Boolean vectors `deleted` and `inqueue` keep track of the vertices that have been deleted from all the graphs (common vertex set V^0) and the vertices that have been added to Q, respectively. For each graph G^i in the sequence, vector $\deg^i$ maps each vertex to its degree in graph G^i (step 5). Once a vertex v is detected to have degree less than k in any graph, it is added to Q if it is not already in G, and `inqueue`$[v]$ is updated to `true` (steps 8 and 18). The `inqueue` entries serve as an indicator that prevents any vertex from being appended to Q more than once. When a vertex v in the front of Q is selected and removed (popped out of Q), the boolean vector `deleted`$[v]$ is updated to `true` for that vertex (step 11), and the degree of each of its remaining neighbors in each graph in the sequence is decreased by one (step 15). At termination, either all vertices are deleted (the persistent k-core is empty), or the surviving vertices correspond to the unique maximal persistent k-core of the graph sequence.

Algorithm 3: (MW-CORE) Find the maximal persistent k-core of a graph sequence

Input: A graph sequence $(G^1, G^2, ..., G^\tau)$ on a common vertex set V^0.
Output: The maximal persistent k-core.

```
1  queue Q ← ∅
2  for v ∈ V^0 do
3      deleted[v] ← false, inqueue[v] ← false
4      for i = 1, 2, ..., τ do
5          deg^i[v] ← |N_{G^i}(v)|
6          if inqueue[v] = false and deg^i[v] < k then
7              Q.push(v)
8              inqueue[v] = true
9  while Q ≠ ∅ do
10     v ← Q.pop()
11     deleted[v] ← true
12     for i = 1, 2, ..., τ do
13         for u ∈ N_{G^i}(v) do
14             if deleted[u] = false then
15                 deg^i[u] ← deg^i[u] − 1
16                 if deg^i[u] < k and inqueue[u] = false then
17                     Q.push(u)
18                     inqueue[u] ← true
19 return {v ∈ V^0 | deleted[v] = false}
```

Steps 1–8 of Algorithm 3 can be completed in $O(\tau|V^0|)$ time. Since each vertex could be appended to Q at most once, the while-loop (line 9) executes at most $|V^0|$ times and each execution takes $O(\tau|N_{G^0}(v)|)$ time. As a result, the entire while-loop can be completed in $O(\tau|E^0|)$ time. Algorithm 3 can therefore be implemented to run in $O(\tau(|V^0| + |E^0|))$ time. By applying Algorithm 3 to each window $\mathcal{G}^{t,\tau}$ in the MW method, we can find a τ-persistent k-core signature of $\mathcal{G}$ in $O(T\tau(|V^0| + |E^0|))$ time.

8 Computational Experiments

The goals of our computational study are twofold: (*i*) to understand the computational gains achieved by using the MW method, with specialized algorithms and with formulation **GSIP-F2-MW**, when compared with directly solving the monolithic formulation **GSIP-F2** using a general purpose MILP solver, and (*ii*) to compare the solutions found by solving GSIP against those obtained by solving the corresponding problem on a single graph. The computational experiments are conducted on 64-bit Linux® compute nodes with dual Intel® "Skylake" 6130 CPUs and 96 GB/768 GB/1.5 TB RAM as needed. All algorithms are implemented in C++ and the MILP formulations are solved using Gurobi™ Optimizer v9.0.1. The C++ implementations of algorithms **MW-CLQ**, **MW-2CLB**, and **MW-CORE** and the test instances used in this computational study are available online (Pan et al., 2020a,b,c).

Based on the results of our preliminary experiments, we disabled all Gurobi default cuts to

achieve better running times. We limit the Gurobi solve-time to two hours per instance when solving **GSIP-F2** formulation (excluding read/write times and model build time). For the MW method, we test performance of the MW method with the generic **GSIP-F2-MW** formulation and with our specialized algorithms (**MW-CLQ**, **MW-2CLB**, and **MW-CORE**). We refer to the MW method in which we solve the generic **GSIP-F2-MW** formulation using Gurobi as **MW-F2**. We impose a one hour time limit for each window, and terminate the algorithm if two consecutive windows reach their time limit. As we do not explicitly limit the overall wall-clock time taken by the MW method, we allow **GSIP-F2** more time for some instances on which the MW method takes longer than two hours; these instances are identified in the individual table notes.

An instance $(\mathcal{G}, \mathcal{P}, \tau)$ for the GSIP consists of a sequence of T graphs. In our experiments we set $T = 10$, $\tau = 3$, and $\mathcal{P}$ is the singleton corresponding to three contrasting graph-theoretic properties, namely, clique, 2-club, and k-core. The computational studies are reported in Sections 8.1, 8.2, and 8.3 based on the discussion in Sections 5, 6, and 7, respectively. In Section 8.4, we test performance of **MW-CLQ**, **MW-2CLB**, and **MW-CORE** algorithms on relatively long graph sequences, and we set $T = 100$ and $\tau = 10$.

For clique signatures, the test bed is generated from the Tenth DIMACS Implementation Challenge (DIMACS-10) benchmark graphs (Bader et al., 2013) and the Second DIMACS Implementation Challenge (DIMACS-2) benchmark graphs (Johnson & Trick, 1996). From DIMACS-10 benchmarks we select 12 graphs and from DIMACS-2 we select 14 to be used as the universal graph $G^0 = (V^0, E^0)$. The DIMACS-2 instances are typically much denser compared to the DIMACS-10 instances, which are all extremely sparse. Both test beds include graphs based on data from a variety of fields. As some of the DIMACS-2 benchmarks are extremely challenging even for the classical maximum clique problem, we selected only instances with fewer than 1000 vertices. The DIMACS-10 and DIMACS-2 instances are summarized in Tables 1 and 2, respectively. The edge density is denoted by ρ and expressed as a percentage in these tables. We create a sequence of graphs $G^t = (V^t, E^t)$ for $t \in [10]$ with identical vertex sets, i.e., $V^t = V^0$ and the edge set E^t is constructed randomly by including an edge in E^0 with probability 0.8.

Table 1: DIMACS-10 graphs

G^0	$\lvert V^0\rvert$	$\lvert E^0\rvert$	$\rho(\%)$
karate	34	78	13.90
lesmis	77	254	8.68
polbooks	105	441	8.08
adjnoun	112	425	6.84
football	115	613	9.35
celegans	453	2025	1.98
email	1133	5451	0.85
polblogs	1490	16715	1.51
netscience	1589	2742	0.22
power	4941	6594	0.05
hep-th	8361	15751	0.05
PGPgiantcompo	10680	24316	0.04

For 2-club signatures, we again use the test bed of DIMACS-10 instances described above. As most DIMACS-2 instances have diameter two, we generate ten additional benchmark graphs using the algorithm described in (Bourjolly et al., 2002). This generator is known to produce hard

Table 2: DIMACS-2 graphs

G^0	$\|V^0\|$	$\|E^0\|$	$\rho(\%)$
C125.9	125	6963	89.85
C250.9	250	27984	89.91
p_hat300-1	300	10933	24.38
p_hat300-2	300	21928	48.89
p_hat300-3	300	33390	74.45
MANN_a27	378	70551	99.01
brock400_2	400	59786	74.92
brock400_4	400	59765	74.89
gen400_p0.9_55	400	71820	90.00
gen400_p0.9_65	400	71820	90.00
gen400_p0.9_75	400	71820	90.00
C500.9	500	112332	90.05
dsjc500_5	500	62624	50.20
dsjc1000-5	1000	249826	50.02

instances for the maximum 2-club problem at an edge density of 10% (Pajouh & Balasundaram, 2012). The ten instances are used as universal graphs and the sequence of ten graphs is generated from each by the same procedure described above. The instances based on the Bourjolly generator are listed in Table 3.

Table 3: Bourjolly-generator graphs

G^0	$\|V^0\|$	$\|E^0\|$	$\rho(\%)$
bg_1	200	2015	10.10
bg_2	200	1983	9.96
bg_3	200	2014	10.10
bg_4	200	1956	9.83
bg_5	200	2033	10.20
bg_6	200	1971	9.90
bg_7	200	2029	10.20
bg_8	200	2037	10.20
bg_9	200	2009	10.10
bg_10	200	1999	10.00

For k-core signatures, we use the DIMACS-10 test bed identified above and selected temporal networks listed in Table 4 from the Stanford Network Analysis Project (SNAP) (Paranjape et al., 2017; Leskovec et al., 2010; Panzarasa et al., 2009). We ignore the orientation of arcs in the directed networks in our test bed. Each edge in these SNAP instances is associated with a timestamp. We order the edges chronologically and then divide these time-stamped edges equally into 10 groups, thus forming the 10-graph sequence.

Table 4: SNAP temporal networks

G^0	$\lvert V^0\rvert$	$\lvert E^0\rvert$	$\rho(\%)$
CollegeMsg	1899	13834	0.77
email-Eu-core-temporal	1005	16064	3.18
sx-askubuntu	515280	455689	0.00034
sx-mathoverflow	88580	187986	0.0048
sx-stackoverflow	6024270	28183518	0.00016
sx-superuser	567315	714565	0.00044
wiki-talk-temporal	1140149	2787966	0.00043

8.1 Results for clique signatures

In this section, we focus on clique signatures and compare the performance of Algorithm 1 (**MW-CLQ**) against solving **MW-F2** directly in the MW method and directly solving the monolithic **GSIP-F2** formulation with constraints (7). In Table 5 we compare the best objective values found by the three methods and the corresponding running times. Out of the 26 instances, **MW-CLQ** reached optimality in 25 instances, while **GSIP-F2** and **MW-F2** did so in 7 and 23 instances, respectively. For those instances on which all three approaches found an optimal solution, running times were orders of magnitude faster with the MW methods compared to solving **GSIP-F2**.

We expect the monolithic formulation solved by a general-purpose solver to simply provide a baseline for comparison and not necessarily be a competitive solver for this problem. However, this baseline serves to highlight the substantial benefits of using the MW method. We can also observe that **MW-CLQ** is significantly faster than **MW-F2** on DIMACS-10 instances and solves all of them to optimality, while the two large instances are not solved to optimality using **MW-F2**. This is because the decomposition and preprocessing techniques used in **MW-CLQ** are very effective on the sparse instances in the DIMACS-10 test bed compared to the DIMACS-2 test bed that contains very dense instances. On the DIMACS-2 test bed the performance of both MW methods are generally comparable, with **MW-CLQ** being faster on most instances solved to optimality by a small margin.

Regarding the objective values on those instances for which the optimal clique and clique signature are found, we note that the optimal signatures found are comparably sized to the largest clique in the snapshot graph G^1. Specifically, the optimal persistent cliques are on average 67.1% the size of the optimal static cliques. It is evident from these results that persistent signatures are non-trivial on our test bed, and possibly more generally in practice.

8.2 Results for 2-club signatures

In this section, we compare the performance of Algorithm 2 (**MW-2CLB**) against solving **MW-F2** directly in the MW method and directly solving the monolithic **GSIP-F2** formulation with common neighbor constraints (10). Recall that **MW-2CLB** uses a decomposition branch-and-cut algorithm on each window, with a 2-clique based master relaxation and a violated common neighbor constraint added as a "lazy" constraint during the progress of the branch-and-cut algorithm. The time limits are the same as those we set for the persistent clique instances.

In Table 6, we present a comparison of the best objectives and the running times for the three approaches. **MW-2CLB** and **MW-F2** solved all 22 instances to optimality, while only 16 were solved to optimality with the **GSIP-F2** formulation. Similar to the previous section, the use of

Table 5: Comparison of best objectives and running times for the maximum 3-persistent clique signature problem on DIMACS-10 and DIMACS-2 instances in our test bed.

		Best objective			Time (s)		
G^0	$\omega(G^1)$	**GSIP-F2**	**MW-CLQ**	**MW-F2**	**GSIP-F2**	**MW-CLQ**	**MW-F2**
karate	4	4	4	4	3.20	0.02	0.16
lesmis	7	5	5	5	27.32	0.04	0.91
polbooks	5	5	5	5	123.32	0.01	2.17
adjnoun	5	5	5	5	157.78	0.01	1.67
football	8	6	6	6	164.45	0.01	1.58
celegans	8	$\geq 4^{a}$	5	5	7287.98	0.14	106.67
email	6	$\geq 5^{a}$	5	5	8087.69	0.48	444.85
polblogs	11	$\geq 6^{a}$	7	7	7240.37	2.23	973.99
netscience	9	$\geq 3^{a}$	6	6	7237.17	0.90	685.26
power	4	$\geq 2^{a}$	4	4	7517.39	1.04	5831.37
hep-th	11	$\geq 2^{ab}$	6	≥ 6	26115.28	16.15	23721.57
PGPgiantcompo	12	$\geq 2^{a}$	8	≥ 1	8888.77	22.71	8003.58
C125.9	16	9	9	9	387.40	5.28	5.72
C250.9	21	≥ 10	11	11	7200.77	170.37	173.38
p_hat300-1	7	6	6	6	5853.01	3.22	53.33
p_hat300-2	13	≥ 8	9	9	7201.49	67.63	74.81
p_hat300-3	17	≥ 10	10	10	7202.14	292.30	296.62
MANN_a27	≥ 28	≥ 12	13	13	7201.34	1767.82	1734.93
brock400_2	16	≥ 9	10	10	7203.05	917.16	975.25
brock400_4	16	≥ 9	10	10	7202.85	911.58	963.44
gen400_p0.9_55	≥ 23	≥ 11	12	12	7201.28	1387.84	1504.08
gen400_p0.9_65	≥ 23	≥ 11	12	12	7201.16	1455.14	1538.17
gen400_p0.9_75	≥ 22	≥ 11	12	12	7201.12	1434.47	1440.79
C500.9	≥ 49	≥ 11	12	12	7204.63	5415.02	5462.18
dsjc500_5	11	≥ 7	8	8	7204.63	1490.54	1538.03
dsjc1000_5	≥ 12	$\geq 8^{abc}$	≥ 9	≥ 9	19649.00	27616.15	7205.68

[a] Memory related crashes with 96 GB RAM and re-run with 768 GB RAM.
[b] **GSIP-F2** was allowed to run longer than two hours based on the running time of MW methods.
[c] **GSIP-F2** was allowed longer than the two hours time limit on this instance because **MW-CLQ** took 27,616.15 seconds. However, memory related crash occurred even with 1.5 TB RAM at 19,649 seconds.
[*] On instances not solved to optimality we report the lower-bound provided by the best solution found.

MW-2CLB, which enables us to decompose the model and employ 2-club-specific preprocessing techniques, improves solution times significantly (by orders of magnitude on DIMACS-10 instances, even compared to **MW-F2**). In addition, the sizes of the largest 2-club signatures on all instances are comparable with the sizes of the maximum 2-club on the static graph G^1, being on average 71.7% the size of their static counterpart. This again suggests that the persistent signatures are non-trivial and detect relevant graph structures that single-snapshot models cannot identify.

8.3 Results for k-core signatures

We report best objective values and running times for computing 3-persistent k-core signatures using the **GSIP-F2** formulation, Algorithm 3 (**MW-CORE**), and **MW-F2** in Table 7. For each instance, we choose k to be the largest integer for which the maximal 3-persistent k-core is not

Table 6: Comparison of best objectives and running times for the maximum 3-persistent 2-club signature problem on DIMACS-10 and Bourjolly-generator instances in our test bed.

		Best objective			Time (s)		
G^0	$\bar{\omega}_2(G^1)$	**GSIP-F2**	**MW-2CLB**	**MW-F2**	**GSIP-F2**	**MW-2CLB**	**MW-F2**
karate	15	14	14	14	0.44	0.02	0.43
lesmis	33	20	20	20	3.19	0.03	1.82
polbooks	21	19	19	19	8.60	0.09	3.03
adjnoun	42	27	27	27	9.84	0.05	3.43
football	14	14	14	14	14.09	0.13	4.50
celegans	188	125	125	125	2663.01	0.59	63.78
email	58	$\geq 31^{a}$	44	44	7218.74	1.43	577.52
polblogs	284	$\geq 178^{b}$	182	182	14435.50	278.84	13582.74
netscience	32	$\geq 26^{a}$	26	26	7247.88	0.43	606.55
power	16	$\geq 4^{a}$	16	16	7533.08	3.01	7494.43
hep-th	44	$\geq 2^{ab}$	38	38	15754.59	9.30	13982.75
PGPgiantcompo	162	$\geq 3^{abc}$	106	106	23219.98	11.11	19826.44
bg_1	32	23	23	23	656.83	42.87	95.14
bg_2	29	22	22	22	586.71	39.53	84.51
bg_3	28	24	24	24	676.66	40.56	83.53
bg_4	27	21	21	21	570.64	40.80	81.80
bg_5	29	26	26	26	648.96	50.07	88.50
bg_6	27	23	23	23	614.94	26.26	84.01
bg_7	31	22	22	22	732.08	49.76	89.49
bg_8	28	22	22	22	698.91	65.59	91.98
bg_9	29	22	22	22	683.03	47.57	94.66
bg_10	32	25	25	25	620.14	31.75	84.15

[a] Memory related crashes with 96 GB RAM and re-run with 768 GB RAM.
[b] **GSIP-F2** was allowed to run longer than two hours based on the running time of MW methods.
[c] Memory related crashes with 768GB RAM and re-run with 1.5 TB RAM.
[*] On instances not solved to optimality we report the lower-bound provided by the best solution found.

empty. Our choice of k is the graph signature analogue of the degeneracy of a (single) graph, which is defined as the largest k for which the given graph contains a non-empty k-core (Szekeres & Wilf, 1968). Although there are alternate approaches available for selecting parameter k, we chose the aforementioned approach to avoid having to make an arbitrary/fixed choice for parameter k across all instances in our test bed.

MW-CORE requires under one second for all but one instance in our test bed, including those instances with more than 100,000 vertices. Being a polynomial-time solvable problem (in fact, linear in the size of the input) the running time is of a different magnitude from what we observe for the clique and 2-club signature problems. **MW-F2** is competitive except on very large instances, especially the five largest SNAP instances. As before, the comparison between the best objective values of GSIP and the size of the maximal k-core of G^1 (for the same k) suggests that the persistent k-core signatures are likely to be non-trivial in practice.

8.4 Results for long sequences

The final experiment of our computational study is to consider signature problems with large values of T. Here, we generate a graph sequence from the instance identified as G^0 using the same

Table 7: Comparison of best objectives and running times for the maximum 3-persistent k-core signature problem on DIMACS-10 and SNAP instances in our test bed.

			Best objective			Time (s)		
G^0	k	core(G^1)	**GSIP-F2**	**MW-CORE**	**MW-F2**	**GSIP-F2**	**MW-CORE**	**MW-F2**
karate	3	11	5	5	5	0.04	0.00	0.06
lesmis	6	13	21	21	21	0.08	0.00	0.11
polbooks	5	45	15	15	15	0.11	0.00	0.10
adjnoun	4	57	50	50	50	0.11	0.00	0.18
football	5	113	114	114	114	0.08	0.00	0.29
celegans	7	53	30	30	30	0.43	0.00	0.30
email	7	278	169	169	169	1.45	0.00	0.81
polblogs	27	102	72	72	72	3.39	0.01	1.04
netscience	12	20	19	19	19	0.91	0.00	0.44
power	3	85	40	40	40	10.27	0.01	0.86
hep-th	15	0	24	24	24	31.55	0.01	1.57
PGPgiantcompo	23	42	42	42	42	78.99	0.02	2.20
CollegeMsg	3	301	45	45	45	1.40	0.00	0.47
email-Eu-core-temporal	10	258	106	106	106	1.75	0.01	0.71
sx-askubuntu	9	1380	≥ 0	16	16	7213.01	0.24	37.88
sx-mathoverflow	13	558	69	69	69	6814.83	0.05	9.11
sx-stackoverflow	26	35119	≥ 0	90	90	7367.62	4.47	596.37
sx-superuser	9	2788	≥ 0	40	40	7215.02	0.25	40.93
wiki-talk-temporal	14	3072	≥ 0	228	228	7230.53	0.56	82.88

* On instances not solved to optimality we report the lower-bound provided by the best solution found.
* The value of parameter k is the largest integer for which a non-empty maximal 3-persistent k-core exists.
* The size of the maximal k-core of G^1 is reported under core(G^1).
* Running time of 0.00 implies that the actual duration was less than 0.005s.

procedure as before, but with $T = 100$ and $\tau = 10$. We test the performance of the specialized MW methods, i.e., **MW-CLQ**, **MW-2CLB**, and **MW-CORE** on DIMACS-10 instances under these settings. Our results for finding 10-persistent clique, 2-club, and k-core signatures (with k chosen as before) are reported in Table 8. These results show that the specialized MW methods remain effective for solving the signature problems even when T and τ take large values.

Table 8: Best objectives and running times for computing 10-persistent graph signatures using algorithms **MW-CLQ**, **MW-2CLB**, and **MW-CORE** on DIMACS-10 instances with $T = 100$.

	Best objective			Time (s)		
G^0	**MW-CLQ**	**MW-2CLB**	**MW-CORE** (k)	**MW-CLQ**	**MW-2CLB**	**MW-CORE**
karate	3	8	18 (2)	0.02	0.62	0.00
lesmis	4	14	12 (6)	0.03	1.40	0.01
polbooks	3	15	38 (4)	0.04	2.38	0.02
adjnoun	3	14	34 (4)	0.08	4.78	0.02
football	3	13	115 (5)	0.07	2.96	0.02
celegans	4	41	22 (7)	0.12	19.92	0.07
email	3	21	199 (6)	5.34	55.07	0.10
polblogs	4	161	87 (26)	6.18	1131.36	0.25
netscience	4	21	20 (12)	0.76	26.39	0.09
power	3	8	13 (3)	2.17	174.87	0.19
hep-th	4	24	24 (14)	8.20	494.94	0.27
PGPgiantcompo	4	65	40 (22)	121.09	1157.27	0.33

9 Concluding Remarks

In many graph-based data mining applications over temporal networks, we are interested in finding subgraphs that persist across a sequence of graphs rather than those found in the individual snapshots in time, or in a graph that aggregates information across time. We introduce the new graph-theoretic paradigm of graph signatures motivated by this observation. Identifying graph signatures could be more valuable in settings where patterns that occur in isolation are not of interest, or if the persistency of a pattern in time indicates its underlying importance in the dynamics of the network.

The proposed concept is not only limited to cluster or community detection challenges in temporal networks, but can serve as a framework for extending any graph-theoretic property or invariant known for a single graph to a sequence of graphs in a straightforward fashion. The framework also facilitates combining multiple properties in a systematic manner when analyzing graph sequences.

In this introductory article, we present mixed-integer programming formulations for the general graph signature identification problem where the properties of interest are mixed-integer representable. We develop a dynamic programming-based algorithm that alleviates the computational challenge by decomposing it into sequential subproblems. Referred to as the moving window (MW) method, the approach further facilitates custom scale-reduction techniques based on decomposition and preprocessing that are specific to the graph property and the mathematical formulation.

Although the monolithic reformulations are effective on small instances, only the MW method adequately scales for larger instances. We find in our computational study that the graph signatures and their static counterparts are comparable in their optimal objective values, suggesting that the optimal graph signatures are non-trivial and can be found with reasonable computational effort in practice. Furthermore, the MW method can be parallelized for very long sequences by partitioning into subsequences, solving each subsequence in a separate thread, and centrally keeping track of the best signature identified by any thread to facilitate preprocessing across all threads. In other words, our framework has the potential to detect important graph signatures in complex dynamic networks for a modest increase in computational effort.

The concept of graph signatures may be extended to stochastic/robust optimization settings, as well as to online optimization settings in which the graph sequence is streaming continuously. Graph properties often used in community detection like clique relaxations (Pattillo et al., 2013) and small-world subgraphs (Kim et al., 2020) could be investigated within the graph signature framework. These would also offer good candidates for extending the MW method (designed for single property signatures in this introductory article) to a disjunction between two related but incomparable properties. Such an approach would entail solving two τ-persistent signature problems in each window in the MW method. For example, the concept behind the small-world subgraph model introduced by Kim et al. (2020) combining local average distance and local clustering coefficient could be revisited in the disjunctive property setting of graph signatures. These are interesting directions to explore in the future.

Acknowledgements

The computing for this project was performed at the High Performance Computing Center at Oklahoma State University supported in part through the National Science Foundation grant OAC-1531128. The research of Juan Borrero was supported in part by the Office of Naval Research Grant ONR–N00014-19-1-2329.

References

Abello, J., Pardalos, P. M., & Resende, M. G. C. (1999). On maximum clique problems in very large graphs. In J. Abello, & J. Vitter (Eds.), *External memory algorithms and visualization* (pp. 119–130). Boston, MA, USA: American Mathematical Society volume 50 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*.

Alba, R. D. (1973). A graph-theoretic definition of a sociometric clique. *Journal of Mathematical Sociology*, *3*, 113–126.

Alhajj, R., & Rokne, J. (Eds.) (2018). *Encyclopedia of Social Network Analysis and Mining*. New York: Springer.

Bader, D. A., Meyerhenke, H., Sanders, P., & Wagner, D. (Eds.) (2013). *Graph partitioning and graph clustering: Tenth* DIMACS *Implementation Challenge Workshop* volume 588 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. Providence, RI: American Mathematical Society.

Balasundaram, B., Butenko, S., & Hicks, I. V. (2011). Clique relaxations in social network analysis: The maximum k-plex problem. *Operations Research*, *59*, 133–142.

Balasundaram, B., Butenko, S., & Trukhanov, S. (2005). Novel approaches for analyzing biological networks. *Journal of Combinatorial Optimization*, *10*, 23–39.

Balasundaram, B., & Pajouh, F. M. (2013). Graph theoretic clique relaxations and applications. In P. M. Pardalos, D.-Z. Du, & R. Graham (Eds.), *Handbook of Combinatorial Optimization* (pp. 1559–1598). New York: Springer. (2nd ed.).

Bentert, M., Himmel, A.-S., Molter, H., Morik, M., Niedermeier, R., & Saitenmacher, R. (2019). Listing all maximal k-plexes in temporal graphs. *ACM Journal of Experimental Algorithmics*, *24*, 1.13:1–1.13:27.

Bomze, I. M., Budinich, M., Pardalos, P. M., & Pelillo, M. (1999). The maximum clique problem. In D.-Z. Du, & P. M. Pardalos (Eds.), *Handbook of Combinatorial Optimization* (pp. 1–74). Dordrecht, The Netherlands: Kluwer Academic Publishers.

Borgatti, S. P., Everett, M. G., & Shirey, P. R. (1990). LS sets, lambda sets, and other cohesive subsets. *Social Networks*, *12*, 337–358.

Bourjolly, J.-M., Laporte, G., & Pesant, G. (2002). An exact algorithm for the maximum k-club problem in an undirected graph. *European Journal Of Operational Research*, *138*, 21–28.

Charikar, M., Naamad, Y., & Wu, J. (2018). On finding dense common subgraphs. `arXiv:1802.06361`.

Cook, D. J., & Holder, L. B. (2006). *Mining graph data*. John Wiley & Sons.

Defense Advanced Research Projects Agency (2011). Graph–theoretic Research in Algorithms and the PHenomenology of Social networks (GRAPHS). Broad Agency Announcement.

Freeman, L. C. (1992). The sociological concept of "group": An empirical test of two models. *American Journal of Sociology*, *98*, 152–166.

Hellmann, T., & Staudigl, M. (2014). Evolution of social networks. *European Journal of Operational Research*, *234*, 583–596.

Himmel, A.-S., Molter, H., Niedermeier, R., & Sorge, M. (2017). Adapting the Bron–Kerbosch algorithm for enumerating maximal cliques in temporal graphs. *Social Network Analysis and Mining*, *7*, 35.

Hu, H., Yan, X., Huang, Y., Han, J., & Zhou, X. J. (2005). Mining coherent dense subgraphs across massive biological networks for functional discovery. *Bioinformatics*, *21*, i213–i221.

Jethava, V., & Beerenwinkel, N. (2015). Finding dense subgraphs in relational graphs. In A. Appice, P. P. Rodrigues, V. Santos Costa, J. Gama, A. Jorge, & C. Soares (Eds.), *Machine Learning and Knowledge Discovery in Databases* (pp. 641–654). Cham: Springer International Publishing.

Jiang, C., Coenen, F., & Zito, M. (2013). A survey of frequent subgraph mining algorithms. *The Knowledge Engineering Review*, *28*, 75–105.

Jiang, D., & Pei, J. (2009). Mining frequent cross-graph quasi-cliques. *ACM Transactions on Knowledge Discovery in Data*, *2*, 16:1–42.

Johnson, D., & Trick, M. (Eds.) (1996). *Cliques, Coloring, and Satisfiablility: Second* DIMACS *Implementation Challenge* volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. Providence, RI: American Mathematical Society.

Junker, B. H., & Schreiber, F. (Eds.) (2008). *Analysis of Biological Networks*. New York: Wiley.

Kelleher, L., & Cozzens, M. (1988). Domination sets in social network graphs. *Mathematical Social Science*, *16*, 267–279.

Kim, J., Veremyev, A., Boginski, V., & Prokopyev, O. A. (2020). On the maximum small-world subgraph problem. *European Journal of Operational Research*, *280*, 818–831.

Kuramochi, M., & Karypis, G. (2005). Finding frequent patterns in a large sparse graph. *Data Mining and Knowledge Discovery*, *11*, 243–271.

Latapy, M., Viard, T., & Magnien, C. (2018). Stream graphs and link streams for the modeling of interactions over time. *Social Network Analysis and Mining*, *8*, 61.

Leskovec, J., Huttenlocher, D., & Kleinberg, J. (2010). Governance in social media: A case study of the wikipedia promotion process. *arXiv preprint arXiv:1004.3547*, .

Li, W., Liu, C.-C., Zhang, T., Li, H., Waterman, M. S., & Zhou, X. J. (2011). Integrative analysis of many weighted co-expression networks using tensor computation. *PLOS Computational Biology*, *7*, 1–13.

Lu, Y., Moradi, E., & Balasundaram, B. (2018). Correction to: Finding a maximum k-club using the k-clique formulation and canonical hypercube cuts. *Optimization Letters*, *12*, 1959–1969.

Luce, R. D. (1950). Connectivity and generalized cliques in sociometric group structure. *Psychometrika*, *15*, 169–190.

Matula, D. W., & Beck, L. L. (1983). Smallest-last ordering and clustering and graph coloring algorithms. *Journal of the ACM*, *30*, 417–427.

Miao, J., & Berleant, D. (2004). From paragraph networks to document networks. In *Proceedings of the International Conference on Information Technology: Coding and Computing, 2004 (ITCC 2004)* (pp. 295–302). volume 1.

Mokken, R. J. (1979). Cliques, clubs and clans. *Quality and Quantity*, *13*, 161–173.

Moradi, E., & Balasundaram, B. (2018). Finding a maximum k-club using the k-clique formulation and canonical hypercube cuts. *Optimization Letters*, *12*, 1947–1957.

Pajouh, F. M., & Balasundaram, B. (2012). On inclusionwise maximal and maximum cardinality k-clubs in graphs. *Discrete Optimization*, *9*, 84–97.

Pan, H., Balasundaram, B., & Borrero, J. S. (2020a). Implementation of the moving window method for the maximum 2-club signature problem. Codes and instances online at: `https://github.com/haonap/2clubSig`.

Pan, H., Balasundaram, B., & Borrero, J. S. (2020b). Implementation of the moving window method for the maximum clique signature problem. Codes and instances online at: `https://github.com/haonap/cliqueSig`.

Pan, H., Balasundaram, B., & Borrero, J. S. (2020c). Implementation of the moving window method for the maximum k-core signature problem. Codes and instances online at: `https://github.com/haonap/kcoreSig`.

Panzarasa, P., Opsahl, T., & Carley, K. M. (2009). Patterns and dynamics of users' behavior and interaction: Network analysis of an online community. *Journal of the American Society for Information Science and Technology*, *60*, 911–932.

Paranjape, A., Benson, A. R., & Leskovec, J. (2017). Motifs in temporal networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining* WSDM '17 (pp. 601–610). New York, NY, USA: Association for Computing Machinery.

Pattillo, J., Youssef, N., & Butenko, S. (2013). On clique relaxation models in network analysis. *European Journal of Operational Research*, *226*, 9–18.

Pei, J., Jiang, D., & Zhang, A. (2005). Mining cross-graph quasi-cliques in gene expression and protein interaction data. In *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on* (pp. 353 – 356).

Rebennack, S., Oswald, M., Theis, D. O., Seitz, H., Reinelt, G., & Pardalos, P. M. (2011). A branch and cut solver for the maximum stable set problem. *Journal of Combinatorial Optimization*, *21*, 434–457.

Salemi, H., & Buchanan, A. (2020). Parsimonious formulations for low-diameter clusters. *Mathematical Programming Computation*, *12*, 493–528.

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

Seidman, S. B., & Foster, B. L. (1978). A graph theoretic generalization of the clique concept. *Journal of Mathematical Sociology*, *6*, 139–154.

Semertzidis, K., Pitoura, E., Terzi, E., & Tsaparas, P. (2019). Finding lasting dense subgraphs. *Data Mining and Knowledge Discovery*, *33*, 1417–1445.

Shahinpour, S., & Butenko, S. (2013). Distance-based clique relaxations in networks: s-clique and s-club. In B. I. Goldengorin, V. A. Kalyagin, & P. M. Pardalos (Eds.), *Models, Algorithms, and Technologies for Network Analysis* (pp. 149–174). Springer New York volume 59.

Szekeres, G., & Wilf, H. S. (1968). An inequality for the chromatic number of a graph. *Journal of Combinatorial Theory*, *4*, 1–3.

Terveen, L., Hill, W., & Amento, B. (1999). Constructing, organizing, and visualizing collections of topically related, web resources. *ACM Transactions on Computer-Human Interaction*, *6*, 67–94.

Veremyev, A., Prokopyev, O. A., & Pasiliao, E. L. (2014). An integer programming framework for critical elements detection in graphs. *Journal of Combinatorial Optimization*, *28*, 233–273.

Verma, A., Buchanan, A., & Butenko, S. (2015). Solving the maximum clique and vertex coloring problems on very large sparse networks. *INFORMS Journal on Computing*, *27*, 164–177.

Viard, T., Latapy, M., & Magnien, C. (2016). Computing maximal cliques in link streams. *Theoretical Computer Science*, *609*, 245–252.

Viard, T., Magnien, C., & Latapy, M. (2018). Enumerating maximal cliques in link streams with durations. *Information Processing Letters*, *133*, 44–48.

Vogiatzis, C., Veremyev, A., Pasiliao, E. L., & Pardalos, P. M. (2015). An integer programming approach for finding the most and the least central cliques. *Optimization Letters*, *9*, 615–633.

Walteros, J. L., & Pardalos, P. M. (2012). Selected topics in critical element detection. In *Applications of Mathematics and Informatics in Military Science* (pp. 9–26). Springer.

Wasserman, S., & Faust, K. (1994). *Social Network Analysis*. New York: Cambridge University Press.

Yan, X., Zhou, X. J., & Han, J. (2005). Mining closed relational graphs with connectivity constraints. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining* KDD '05 (pp. 324–333). New York, NY, USA: ACM.