On Atomic Cliques in Temporal Graphs

Yajun Lu

Department of Management & Marketing, Jacksonville State University, United States

Zhuqi Miao

State University of New York at New Paltz, United States

Parisa Sahraeian, Balabhaskar Balasundaram*

School of Industrial Engineering & Management, Oklahoma State University, United States

Abstract

Atomic cliques were introduced recently to analyze comorbidity graphs that vary over time. We consider the atomic counterpart of the classical maximum clique problem in this paper. Our main contribution is a polynomial-time algorithm that transforms the maximum atomic clique problem to the maximum clique problem on an auxiliary graph. We report results from our computational studies that demonstrate the effectiveness of this transformation in solving the maximum atomic clique problem in comparison to direct integer programming based approaches.

Keywords: clique, temporal graph, atomic, indivisible, unsplittable

1. Introduction

A subset of pairwise adjacent vertices in a graph is called a *clique*. The *maximum clique problem* (MCP) seeks to find a clique of maximum cardinality in the given graph. This classical combinatorial optimization problem has numerous applications [4], because cliques are idealized representations of "tightly-knit" clusters in the graph model. Cliques for instance were considered to be ideal representations of cohesive social subgroups in social network analysis [20].

Many extensions of the clique model have been introduced recently in the temporal graph setting in which the vertices and/or edges are assumed to change over time (e.g., Δ-*cliques* [9, 18] and *clique signatures* [3]). We are interested in *atomic cliques* in this paper, a notion recently introduced to cluster and analyze temporal comorbidity graphs [12]. Informally, an atomic clique is a clique that is unsplittable[1] over time—the clique is either present or absent entirely and no parts of it appears in the temporal graph unless the entire clique is present. One could therefore interpret atomicity as an extension of the cohesiveness of a clique in the temporal dimension, a requirement that enables us to study the progression of a cohesive subgroup over time.

Consider a collection of graphs $\mathcal{G}$, which if we assume are indexed by discrete points in time, can represent a graph that is evolving over time. Denote the vertex set of each graph $G \in \mathcal{G}$ by $V(G)$ and the edge set by $E(G)$. We refer to an edge $\{u, v\} \in E(G)$ using the short form uv. We denote by G^0 the *support graph* of the collection $\mathcal{G}$; by which we mean each graph $G \in \mathcal{G}$ is a subgraph of G^0. Without loss of generality, we assume G^0 to be minimal by exclusion of vertices and edges.

Definition 1. A subset of vertices $S \subseteq V(G^0)$ is called an *atomic clique* if one of the following conditions hold in every graph $G \in \mathcal{G}$:

1. $S \subseteq V(G)$ and forms a clique in G, or
2. $S \cap V(G) = \emptyset$.

Observe that the first condition ensures that if any vertex from the atomic clique is present in an arbitrary graph from the collection, the entire subset is present and forms a clique in that graph. (Implicit in that observation is the fact that the vertex sets of the graphs in the collection $\mathcal{G}$ are not assumed to be identical.) If the first condition does not hold, then the second condition must—the atomic clique is entirely absent from that graph. Figure 1 is a collection $\mathcal{G}$ of three graphs and Figure 2 shows four atomic cliques that partition $V(G^0)$.

Lu et al. [12] introduced the notion of atomic cliques and presented an integer programming (IP) based heuris-

*Corresponding author

Email addresses: ylu@jsu.edu (Yajun Lu), miaoz@newpaltz.edu (Zhuqi Miao), parisa.sahraeian@okstate.edu (Parisa Sahraeian), baski@okstate.edu (Balabhaskar Balasundaram)

[1] Much to the chagrin of particle physicists, we are sticking with this terminology introduced in [12] because "bosonic" clique does not have quite the same ring!

tic to partition V^0 into atomic cliques. These partitions were subsequently used to cluster and visualize temporal comorbidity graphs to help recognize disease progression and comorbidity in clinical decision support systems.

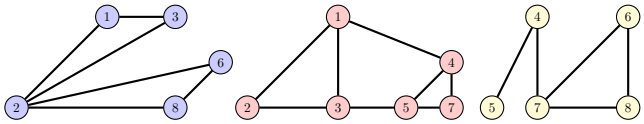

Figure 1: Collection $\mathcal{G}$

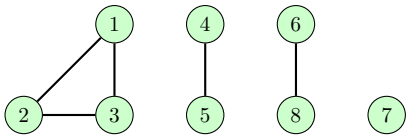

Figure 2: Atomic cliques in the graph collection in Figure 1

Our main result presented in Section 2 is an "edge peeling" algorithm that constructs an auxiliary graph $\widehat{G}$ in polynomial time, with the property that S is an atomic clique in $\mathcal{G}$ if and only if S is a clique in $\widehat{G}$. Consequently, the *maximum atomic clique problem* (MACP) to find an atomic clique of maximum cardinality in $\mathcal{G}$, and the *minimum atomic clique partition problem* to partition $V(G^0)$ into a minimum number of atomic cliques can both be reduced to solving their classical (NP-hard) counterparts on $\widehat{G}$.

To assess the usefulness of our algorithm in practice, we introduce an IP formulation of the MACP in Section 3 including reformulations to reduce the number of nonzero coefficients in the constraints. In Section 4, we compare the performance of a commercial IP solver in solving this formulation against solving the MCP on $\widehat{G}$ using a recent maximum clique solver [19]. Our codes and instances are shared publicly on GitHub at `https://github.com/yajun668/AtomicCliques`. We discuss the limitations to extending our approach to more general clique relaxations in Section 5 and the paper is concluded in Section 6.

2. Edge peeling

The transformation of atomic cliques in $\mathcal{G}$ to cliques in an auxiliary graph $\widehat{G}$ begins with the following key observation. Consider an edge that is present in graph $G \in \mathcal{G}$ and not in graph $H \in \mathcal{G}$. If this is because $V(H)$ contains exactly one of the endpoints or because $V(H)$ contains both endpoints but they are not adjacent in H, then including both endpoints will violate the conditions of Definition 1 for graph H. This observation leads us to Lemma 1, which states that such an edge can be deleted without impacting any of the atomic cliques in $\mathcal{G}$. Algorithm 1 recursively deletes such edges until none are left.

Lemma 1. *Let $\mathcal{G}'$ be the input to Algorithm 1 and $\mathcal{G}$ its output. Then, S is an atomic clique in $\mathcal{G}$ if and only if it is an atomic clique in $\mathcal{G}'$.*

Algorithm 1: Edge Peeling: Generic Version for Atomic Cliques.

Input: $\mathcal{G}$

while $\exists uv \in E(G) \setminus E(H)$ *for some* $G, H \in \mathcal{G}$ *and* $V(H) \cap \{u, v\} \neq \emptyset$ **do**

  delete edge uv from every graph that contains it

return $\mathcal{G}$

After Algorithm 1 "peels" the collection of graphs, the resulting collection satisfies an interesting property that takes us a step closer to the auxiliary graph $\widehat{G}$. Henceforth, we use $\mathrm{cc}(G)$ to denote the collection of (maximal) connected components of graph G.

Lemma 2. *Algorithm 1 produces a consistent set of connected components after edge peeling. That is, if $J \in \mathrm{cc}(G)$ and $K \in \mathrm{cc}(H)$ for graphs $G, H \in \mathcal{G}$, where $\mathcal{G}$ is the output of Algorithm 1, then one of the following conditions holds:*

1. *Either $V(J) \cap V(K) = \emptyset$; or,*
2. *J and K are identical graphs, i.e., $V(J) = V(K)$ and $E(J) = E(K)$.*

Proof. Note that one of the conditions will trivially hold if J and K belong to the same graph from the collection (i.e., $G = H$). Now consider the nontrivial case where the graphs G and H are distinct members of the collection $\mathcal{G}$. To arrive at a contradiction, we suppose that $V(J) \neq V(K)$ and $V(J) \cap V(K) \neq \emptyset$.

Then, we can assume without loss of generality that there exist vertices $u \in V(J) \setminus V(K)$ and $v \in V(J) \cap V(K)$. Consider a path that connects vertices u and v in J. Let vertex a be the last vertex on this path that does not belong to K (it is possible $u = a$). We let the next vertex on this path be b (it is possible $b = v$). Note that vertex $b \in V(J) \cap V(K)$ and $ab \in E(J)$. However, ab is not an edge in K because vertex $a \notin V(K)$. This leads to a contradiction as the edge ab cannot survive the edge peeling algorithm because $ab \in E(G) \setminus E(H)$ and $V(H) \cap \{a, b\} \neq \emptyset$.

To show that $E(J) = E(K)$ given that $V(J) = V(K)$, assume without loss of generality that an edge uv exists in J but not in K. Note that the end points u and v are also present in K. That means $uv \in E(G) \setminus E(H)$ and $V(H) \cap \{u, v\} \neq \emptyset$ contradicting the output condition of the edge peeling algorithm. □

Theorem 1. *Let $\mathcal{G}'$ and $\mathcal{G}$ be the input and output of Algorithm 1, respectively. Let $\widehat{G}$ be the (auxiliary) graph whose connected components are precisely the union of the consistent set of connected components of the graphs in the collection $\mathcal{G}$. In other words, we let $V(\widehat{G}) \coloneqq \bigcup_{G \in \mathcal{G}} V(G)$ and $E(\widehat{G}) \coloneqq \bigcup_{G \in \mathcal{G}} E(G)$. Then, S is an atomic clique of $\mathcal{G}'$ if and only S is a clique of $\widehat{G}$.*

Proof. If S is an atomic clique in $\mathcal{G}$, then by definition every edge that exists between vertices in S will survive the edge peeling Algorithm 1. Hence, S must form a clique in $\widehat{G}$.

Conversely, let S be a clique in $\widehat{G}$. It must be contained within a connected component J of $\widehat{G}$. Component J must also belong to the consistent connected components produced by edge peeling. Hence, based on Lemma 2, $S \subseteq V(G)$ for every graph G that contained the connected component J, wherein S forms a clique; and $S \cap V(G) = \emptyset$ for every graph G that does not contain J. Then, S is an atomic clique in $\mathcal{G}$ output by the edge peeling Algorithm 1. Moreover, by Lemma 1, S is an atomic clique in the input graph collection $\mathcal{G}'$ if and only if S is a clique in $\widehat{G}$. □

Theorem 1 establishes our main result showing that atomic cliques are preserved as cliques by the edge peeling algorithm that produces a consistent set of connected components, which can essentially be treated as a single auxiliary graph. Consequently, the MACP can be transformed to the classical MCP on the auxiliary graph $\widehat{G}$.

Before concluding this section we present a more detailed pseudocode for edge peeling in Algorithm 2. This version is similar to our implementation that, instead of iteratively deleting edges, collects edges that will survive the edge peeling conditions of Algorithm 1. Specifically, we preserve an edge $uv \in E(G^0)$ if for each $G \in \mathcal{G}$, either $uv \in E(G)$ or $V(G) \cap \{u, v\} = \emptyset$. Algorithm 2 can be implemented to run in $\mathcal{O}(p(m+n))$ time, where p is the number of graphs in the collection, $m = |E(G^0)|$, and $n = |V(G^0)|$.

Algorithm 2: Edge Peeling for Atomic Cliques.

Input: $\mathcal{G}$

Construct support graph G^0
$\mathcal{I}(v) \leftarrow \{G \in \mathcal{G} : v \in V(G)\}$ $\quad \forall v \in V(G^0)$
$\mathcal{J}(uv) \leftarrow \{G \in \mathcal{G} : uv \in E(G)\}$ $\quad \forall uv \in E(G^0)$
contain$[v, G] \leftarrow$ *false* $\quad \forall v \in V(G^0), G \in \mathcal{G}$
for $v \in V(G^0)$ **do**
  for $G \in \mathcal{I}(v)$ **do**
    contain$[v, G] \leftarrow$ *true*
$V(\widehat{G}) \leftarrow V(G^0), E(\widehat{G}) \leftarrow \emptyset$
for $uv \in E(G^0)$ **do**
  preserve $\leftarrow$ *true*
  contain-edge$[G] \leftarrow$ *false* $\quad \forall G \in \mathcal{G}$
  contain-edge$[G] \leftarrow$ *true* $\quad \forall G \in \mathcal{J}(uv)$
  for $G \in \mathcal{G}$ **do**
    if contain-edge$[G]$ = *false* **then**
      if contain$[u, G]$ = *true or* contain$[v, G]$ = *true* **then**
        preserve $\leftarrow$ *false*
        break
  if *preserve* = *true* **then**
    $E(\widehat{G}) \leftarrow E(\widehat{G}) \cup \{uv\}$
return $\widehat{G}$

3. Formulation and refinements

Although an IP approach was used in the heuristic developed in [12] to partition $V(G^0)$ into atomic cliques, their IP did not explicitly formulate atomic cliques. In formulation (1) of the MACP that we introduce next, we use decision variables $x_u \in \{0, 1\}$ to indicate membership of each vertex $u \in V(G^0)$ in the atomic clique. The complement graph of G is denoted by $\overline{G}$.

$$\max \sum_{u \in V(G^0)} x_u \tag{1a}$$

$$x_u + x_v \leq 1 \quad \forall uv \in E(\overline{G}), G \in \mathcal{G} \tag{1b}$$

$$x_u + x_v \leq 1 \quad \forall u \in V(G), v \notin V(G), G \in \mathcal{G} \tag{1c}$$

$$x_u \in \{0, 1\} \quad \forall u \in V(G^0) \tag{1d}$$

Objective function (1a) maximizes the size of the chosen vertex subset $S \coloneqq \{u \in V(G^0) : x_u = 1\}$ and the clique constraints (1b) ensure that nonadjacent vertex pairs in any graph are not simultaneously included in S. Atomicity constraints (1c) ensure that the clique is unsplittable, i.e., either $S \subseteq V(G)$ or $S \subseteq V(G^0) \setminus V(G)$.

Proposition 1. *Formulation* (1) *is correct.*

Proof. If $S \subseteq V(G^0)$ is an atomic clique, it is easy to verify that its incidence vector x is feasible to formulation (1). Conversely, let x be a feasible solution to formulation (1) and let $S \coloneqq \{u \in V(G^0) : x_u = 1\}$. For a graph $G \in \mathcal{G}$ chosen arbitrarily, consider the nontrivial case $S \cap V(G) \neq \emptyset$. As x satisfies the clique constraints (1b), $S \cap V(G)$ forms a clique in G. If $S \setminus V(G) \neq \emptyset$, then atomicity constraints (1c) corresponding to each $v \in S \setminus V(G)$ and $u \in S \cap V(G)$ will be violated. Hence, S is contained within $V(G)$ and forms an atomic clique as G is arbitrary. □

Two simple tricks can be used to reduce the number of nonzero coefficients in formulation (1) by introducing two new sets of variables. Reducing the number of nonzeroes typically has beneficial effects when using general purpose IP solvers to solve the MCP [14].

For each $G \in \mathcal{G}$ and connected component $J \in \text{cc}(G)$, we introduce the binary variable y_J^G, which must be at one if a vertex from that component is included in the atomic clique. Constraint (1b) can be replaced with the following constraints.

$$\sum_{J \in \mathrm{cc}(G)} y_J^G \leq 1 \qquad \forall G \in \mathcal{G} \quad (2a)$$

$$x_u \leq y_J^G \qquad \forall u \in V(J), J \in \mathrm{cc}(G), G \in \mathcal{G} \quad (2b)$$

$$x_u + x_v \leq 1 \qquad \forall uv \in E(\overline{J}), J \in \mathrm{cc}(G), G \in \mathcal{G} \quad (2c)$$

$$y_J^G \in \{0,1\} \qquad \forall J \in \mathrm{cc}(G), G \in \mathcal{G} \quad (2d)$$

Constraints (2a) ensure that the atomic clique is contained within at most one connected component for each graph, and in conjunction, constraint (2b) forces the variables to zero for vertices in a component that does not contain the atomic clique. The clique constraints (2c) can now be written just for the nonadjacent vertex pairs inside a connected component.

Next, we introduce binary variable z_G for each $G \in \mathcal{G}$ to sparsify the atomicity constraints (1c) by recognizing which graphs actually contain the atomic clique. If z_G is one indicating the presence of the atomic clique, we force variables corresponding to vertices in $V(G^0) \setminus V(G)$ to zero in constraint (3a). Similarly, if z_G is zero, we force the vertices in that graph to be excluded using constraints (3b). We can now replace atomicity constraints (1c) by the following.

$$x_v \leq 1 - z_G \qquad \forall v \notin V(G), G \in \mathcal{G} \quad (3a)$$

$$x_u \leq z_G \qquad \forall u \in V(G), G \in \mathcal{G} \quad (3b)$$

$$z_G \in \{0,1\} \qquad \forall G \in \mathcal{G} \quad (3c)$$

4. Computational study

The objective of our computational study is to gauge the effectiveness of edge peeling in conjunction with an MCP solver in solving the MACP. As our MCP solver, we use the implementation of Walteros and Buchanan [19], henceforth called the *WB solver*, which is a state-of-the-art solver for the MCP. As a baseline for comparison, we solve the enhanced formulation (1) using a general purpose IP solver. The enhanced formulation (EF) for the MACP is obtained by replacing constraints (1b) and (1c) in formulation (1) with the sparser constraints (2) and (3), respectively. We solve the EF for the MACP using the Gurobi Optimization Solver [8] version 9.5.1 in its default settings (which uses multiple threads when available). Our test bed consists of graph collections generated from DIMACS Clique Challenge benchmarks [10] and real-life temporal graph collections from the Stanford Large Network Dataset Collection (SNAP) [11].

Given a seed graph $G^s = (V^s, E^s)$ and the desired number of graphs p, three user-specified parameters q_1, q_2, and q_3 are used to control the addition/deletion of edges and addition of new vertices (along with their incident edges) in the generator described in Algorithm 3. Note that $\binom{V^s}{2}$ is used to denote the set of all possible edges

Algorithm 3: Test Bed Generation.

Input: a seed graph $G^s = (V^s, E^s)$; number of graphs p in the collection $\mathcal{G}$; $q_1, q_2, q_3 \in (0,1)$

```
while |𝒢| < p do
    V ← V^s; E ← ∅; E' ← ∅; E'' ← ∅
    for each e ∈ C(V^s, 2) \ E^s do
        add e to E^+ with probability q_1
    for each e ∈ E^s do
        add e to E^- with probability q_2
    E ← (E^s \ E^-) ∪ E^+
    Calculate the current edge density
      d ← 2|E|/[|V| × (|V| − 1)]
    Create new vertices V' such that |V'| = q_3|V|
    for each e ∈ C(V ∪ V', 2) \ C(V, 2) do
        add e to E' with probability d
    V ← V ∪ V'; E ← E ∪ E'
    G ← (V, E)
    𝒢 ← 𝒢 ∪ G
return 𝒢
```

on V^s. As our seed graphs, we select several graphs from the Second DIMACS Implementation Challenge [10] that are frequently used as benchmarks for the maximum clique problem. From each seed graph, we generate $p = 10$ graphs with probabilities $q_1 = q_3 = 0.1$ and $q_2 = 0.05$.

Table 1 reports the following basic information for each instance: the number of vertices in the support graph ($|V(G^0)|$), number of graphs in the graph collection, total number of edges in the collection ($\sum_{G \in \mathcal{G}} |E(G)|$), the number of edges in the auxiliary graph ($|E(\widehat{G})|$) generated by the edge peeling Algorithm 2, and the maximum atomic clique number. Table 1 also reports the wall-clock running times excluding read/write times using the two solvers: "EP+WB" is the proposed approach that applies the edge peeling algorithm to $\mathcal{G}$ followed by the WB solver on $\widehat{G}$; "EF" column reports the total time to build the model based on the enhanced formulation and solving it using the Gurobi solver in its default settings. Terminal optimality gap (%) is reported for instances that time out after one hour.

As expected, the proposed approach combining edge peeling with a solver for the MCP significantly outperforms solving the enhanced formulation directly using the Gurobi solver. Figure 3 illustrates performance profiles [6, 7] based on the wall-clock times comparing the two solvers for the MACP across all instances in Table 1. For each solver i we plot $f_i(\tau)$, the fraction of the instances in Table 1 for which the running time required by solver i is at most a factor τ of the running time of the fastest solver for that instance. Following convention, we take the solution time to be equal to the time limit for instances that timed out [6]. The performance profiles show that the solver

EP+WB consistently outperforms EF. We can also see from Table 1 that EP+WB can solve several challenging instances in approximately 100 seconds, while EF times out after one hour.

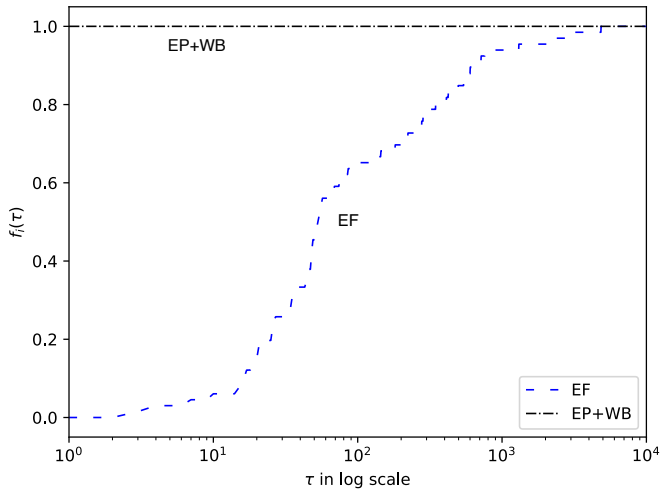

Figure 3: Performance profile comparing the two approaches for solving the maximum atomic clique problem.

Next we consider five real-life temporal graphs from the SNAP database documenting time-stamped interactions in various online fora over a span of more than 2300 days. Edges corresponding to interactions that happened in the same calendar year were grouped into the same graph. We also consider a relatively small temporal network called *CollegeMsg*, which represents private messages sent on an online social network at the University of California, Irvine over a span of 193 days. Each graph in this case represents interactions from the same month. As before, basic information for each of these instances is outlined in Table 2. Solver EP+WB solves all of these instances in under 38 seconds (all except one instance were solved in under 4 seconds), while EF only solves the smallest instance *CollegeMsg* in 116.46 seconds. The results again show that the proposed approach is very effective even on large-scale real-life social networks.

5. Atomic counterparts of other clique relaxations

A natural question that follows is whether the proposed polynomial-time transformation can be extended to the atomic counterparts of clique relaxations like k-plex [2], s-club [1, 5], quasi-clique [13, 16], among others [17]. Recall that central to proving our main result in Theorem 1 is Lemma 2 that shows that edge peeling produces a consistent set of connected components, which in turn allows us to "eliminate duplicates" and treat the collection as a single (auxiliary) graph.

By contrast, defining properties of clique relaxations may not allow us to delete edges as we do in Algorithm 1. For instance, consider 2-clubs (a vertex subset in which every pair of vertices are either adjacent or have a common neighbor in the subset) with atomicity requirements. In the graphs in Figure 4, edges $\{2,3\}$ and $\{2,4\}$, which are present in one graph and not the other, cannot be deleted because subset $\{2,3,4\}$ is an atomic 2-club that will be destroyed if either edge is deleted. This is not only due to the 2-club property not being hereditary (i.e., a property preserved under vertex deletion). The same counterexample works for the hereditary 2-plex property, which limits every vertex in the 2-plex to at most one non-neighbor, as the subset $\{2,3,4\}$ is also an atomic 2-plex.

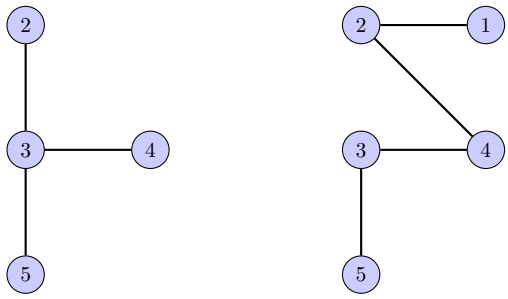

Figure 4: Edges $\{2,3\}$ and $\{2,4\}$ can be deleted when finding atomic cliques, but not when finding atomic 2-clubs or 2-plexes. Although the edge $\{1,2\}$ in the graph on the right can be deleted based on the atomicity property, the result is not a consistent set of connected components because the two connected components comprising of vertices $\{2,3,4,5\}$ on the left and right graphs are not identical.

However, we can still delete an edge $uv \in E(G) \setminus E(H)$ for $G, H \in \mathcal{G}$ such that $|V(H) \cap \{u,v\}| = 1$. On the basis of atomicity alone, no matter the clique relaxation, vertices u and v cannot be included simultaneously. This observation could still be useful as a preprocessing technique. In fact, in Lemma 3 we claim a stronger property for the output of Algorithm 4, which formalizes this limited version of edge peeling. The proof of Lemma 3 is essentially identical to the proof of Lemma 2 except for the final paragraph arguing for identical edge sets, and is therefore omitted.

Algorithm 4: Atomicity-based Edge Peeling for Clique Relaxations.

Input: $\mathcal{G}$

while $\exists uv \in E(G) \setminus E(H)$ *for some* $G, H \in \mathcal{G}$ *and* $|V(H) \cap \{u,v\}| = 1$ **do**
  delete edge uv from every graph that contains it

return $\mathcal{G}$

Lemma 3. *Let $\mathcal{G}$ be the output of Algorithm 4. For graphs $G, H \in \mathcal{G}$, if $J \in \mathrm{cc}(G)$ and $K \in \mathrm{cc}(H)$ then their vertex sets $V(J)$ and $V(K)$ are either disjoint, or identical.*

Consider Figure 4 again as an example. After edge $\{1,2\}$ on the right graph is removed on the basis of atomicity, it yields connected components induced by the same vertex set $\{2,3,4,5\}$ in the two graphs. The singleton $\{1\}$ is the other connected component in the graph on the

right. The maximum 2-club in the graph on the left is $\{2, 3, 4, 5\}$, while the graph on the the right has two maximum size 2-clubs $\{2, 3, 4\}$ and $\{3, 4, 5\}$. The maximum atomic 2-clubs are $\{2, 3, 4\}$ and $\{3, 4, 5\}$.

This interesting observation from Lemma 3 may be used to solve a maximum atomic 2-club problem by transforming it to a maximum cross-graph 2-club problem [15]. Given a collection of graphs $\mathcal{G}$, the maximum cross-graph 2-club seeks to find a subset of vertices $S \subseteq V(G)$ for all $G \in \mathcal{G}$ that forms a 2-club in every $G \in \mathcal{G}$. The upshot is that atomic counterparts of clique relaxations are still very much open for focused studies in the future.

6. Conclusion

We study atomic cliques, a novel concept introduced recently to analyze disease progression in temporal comorbidity graphs, by clustering the temporal graph into unsplittable cliques. We design a purely graph-theoretic algorithm that runs in polynomial time and reduces atomic cliques in a graph collection to classical cliques in an auxiliary graph. Our computational study exploiting this transformation demonstrates the viability of this approach in finding a maximum cardinality atomic clique in a test bed of instances including synthetic benchmarks and large-scale real-life social networks. The approach we introduce is also applicable when solving variants like the minimum atomic clique partitioning problem or the maximum weighted atomic clique problem. However, we also demonstrate that the consistent connected components property that is central to our transformation does not hold in general for the atomic counterparts of graph-theoretic clique relaxations. These variants will require the development of IP approaches for solving appropriate extensions of our IP formulation. The atomicity constraints and their reformulation remain valid for other clique relaxations as well. Alternatively, one could exploit Lemma 3 to decompose the graphs in the collection and solving the cross-graph problem instead, which is also a problem that merits further attention on its own right.

Acknowledgments

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 authors would also like to thank José Walteros and Austin Buchanan for making their solver publicly available [19].

References

[1] B. Balasundaram, S. Butenko, and S. Trukhanov. Novel approaches for analyzing biological networks. *Journal of Combinatorial Optimization*, 10(1):23–39, August 2005.

[2] B. Balasundaram, S. Butenko, and I. V. Hicks. Clique relaxations in social network analysis: The maximum k-plex problem. *Operations Research*, 59(1):133–142, January-February 2011.

[3] B. Balasundaram, J. S. Borrero, and H. Pan. Graph signatures: Identification and optimization. *European Journal of Operational Research*, 296(3):764–775, February 2022.

[4] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo. The maximum clique problem. In D.-Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization*, pages 1–74. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

[5] J.-M. Bourjolly, G. Laporte, and G. Pesant. Heuristics for finding k-clubs in an undirected graph. *Computers & Operations Research*, 27:559–569, 2000.

[6] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91 (2):201–213, 2002.

[7] N. Gould and J. Scott. A note on performance profiles for benchmarking software. *ACM Transactions on Mathematical Software (TOMS)*, 43(2):1–5, 2016.

[8] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022. URL `http://www.gurobi.com`.

[9] A.-S. Himmel, H. Molter, R. Niedermeier, and M. Sorge. Adapting the Bron–Kerbosch algorithm for enumerating maximal cliques in temporal graphs. *Social Network Analysis and Mining*, 7(1):35, Jul 2017.

[10] D. Johnson and M. Trick, editors. *Cliques, Coloring, and Satisfiablility: Second* Dimacs *Implementation Challenge*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, Providence, RI, 1996.

[11] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[12] Y. Lu, S. Chen, Z. Miao, D. Delen, and A. Gin. Clustering temporal disease networks to assist clinical decision support systems in visual analytics of comorbidity progression. *Decision Support Systems*, 148:113583, 2021.

[13] Z. Miao and B. Balasundaram. An ellipsoidal bounding scheme for the quasi-clique number of a graph. *INFORMS Journal on Computing*, 32(3):763–778, 2020. Codes/instances online at: `https://github.com/baski363/Quasi-clique`.

[14] M. J. Naderi, A. Buchanan, and J. L. Walteros. Worst-case analysis of clique MIPs. *Mathematical Programming*, 2021. doi: 10.1007/s10107-021-01706-2.

[15] H. Pan, B. Balasundaram, and J. S. Borrero. A decomposition branch-and-cut algorithm for the maximum cross-graph k-club problem. In *Proceedings of the 10th International Network Optimization Conference (INOC)*, pages 17–22. Open Proceedings, 2022. URL `http://www.openproceedings.org/html/pages/2022_inoc.html`.

[16] J. Pattillo, A. Veremyev, S. Butenko, and V. Boginski. On the maximum quasi-clique problem. *Discrete Applied Mathematics*, 161(1–2):244–257, 2013.

[17] J. Pattillo, N. Youssef, and S. Butenko. On clique relaxation models in network analysis. *European Journal of Operational Research*, 226(1):9–18, 2013.

[18] T. Viard, M. Latapy, and C. Magnien. Computing maximal cliques in link streams. *Theoretical Computer Science*, 609: 245–252, 2016.

[19] J. L. Walteros and A. Buchanan. Why is maximum clique often easy in practice? *Operations Research*, 68(6):1866–1895, 2020.

[20] S. Wasserman and K. Faust. *Social Network Analysis*. Cambridge University Press, New York, 1994.

Table 1: Wall-clock running times (in seconds) observed using the EP+WB and EF solvers for the maximum atomic clique problem on the DIMACS derived test bed. Optimality gap is reported for instances not solved to optimality under the 1-hour time limit.

Name	$\lvert V(G^0)\rvert$	$\lvert\mathcal{G}\rvert$	$\sum_{G\in\mathcal{G}}\lvert E(G)\rvert$	$\lvert E(\widehat{G})\rvert$	Objective	Wall-clock time EP+WB	EF
johnson8-2-4_new	30	10	2,419	121	4	0.01	0.17
MANN-a9_new	49	10	9,761	536	8	0.02	0.08
hamming6-2_new	70	10	19,622	1,111	9	0.03	0.21
hamming6-4_new	70	10	10,074	423	4	0.01	0.26
johnson8-4-4_new	77	10	20,843	1,105	7	0.03	0.30
johnson16-2-4_new	132	10	61,346	3,228	7	0.07	1.44
C125-9_new	137	10	75,134	4,124	10	0.14	2.36
keller4_new	188	10	110,724	5,666	8	0.13	3.39
c-fat200-2_new	220	10	69,258	1,960	8	0.02	1.72
c-fat200-1_new	220	10	54,883	930	6	0.01	1.82
c-fat200-5_new	220	10	113,893	5,045	11	0.06	4.92
san200-0-7-2_new	220	10	160,392	8,353	10	0.25	6.30
sanr200-0-7_new	220	10	159,977	8,326	9	0.27	6.85
brock200-1_new	220	10	167,918	8,889	10	0.36	7.23
san200-0-7-1_new	220	10	160,472	8,419	11	0.21	7.23
gen200-p0-9-55_new	220	10	194,156	10,734	12	0.53	8.02
brock200-3_new	220	10	144,510	7,294	8	0.18	8.13
brock200-4_new	220	10	153,364	7,842	9	0.21	9.12
san200-0-9-2_new	220	10	194,294	10,725	11	0.53	27.91
san200-0-9-1_new	220	10	194,283	10,692	12	0.47	31.08
gen200-p0-9-44_new	220	10	194,068	10,729	12	0.61	32.86
sanr200-0-9_new	220	10	193,705	10,683	11	0.95	33.37
san200-0-9-3_new	220	10	194,084	10,712	12	0.61	34.59
C250-9_new	275	10	303,715	16,844	13	2.09	66.64
hamming8-4_new	281	10	245,217	12,541	9	0.28	15.38
hamming8-2_new	281	10	336,498	18,902	14	2.96	201.67
p-hat300-3_new	330	10	378,502	20,088	11	0.93	134.63
MANN-a27_new	415	10	748,048	42,419	17	48.49	981.03
san400-0-5-1_new	440	10	506,699	23,906	8	1.03	55.94
san400-0-7-3_new	440	10	642,670	33,300	12	2.46	324.57
sanr400-0-5_new	440	10	507,654	24,000	8	0.62	471.30
sanr400-0-7_new	440	10	643,225	33,656	11	3.07	851.87
gen400-p0-9-75_new	440	10	778,245	43,148	15	21.87	976.20
gen400-p0-9-65_new	440	10	778,464	42,948	14	21.30	1015.70
brock400-1_new	440	10	675,863	36,001	11	4.76	1021.22
san400-0-9-1_new	440	10	778,475	43,129	14	22.62	1064.99
gen400-p0-9-55_new	440	10	778,041	42,807	14	21.45	1077.59
san400-0-7-1_new	440	10	643,322	33,498	13	2.56	1083.69
brock400-2_new	440	10	676,801	35,966	11	4.71	1211.50
brock400-3_new	440	10	675,046	35,689	11	4.87	1379.54
brock400-4_new	440	10	675,461	35,768	11	4.63	1462.67
san400-0-7-2_new	440	10	642,531	33,499	12	3.59	1474.75
johnson32-2-4_new	545	10	1,173,263	64,445	13	70.53	1023.76
c-fat500-1_new	550	10	301,027	2,747	7	0.05	17.34
c-fat500-2_new	550	10	340,853	5,497	8	0.06	36.32
c-fat500-5_new	550	10	459,822	13,967	11	0.12	72.09
p-hat500-1_new	550	10	531,330	19,097	7	0.13	169.64
c-fat500-10_new	550	10	659,715	27,992	13	0.51	276.11
DSJC500-5_new	550	10	794,731	37,269	8	1.47	1050.81
p-hat500-2_new	550	10	797,632	37,559	11	1.21	2714.50
p-hat500-3_new	550	10	1,059,702	56,210	13	13.71	3050.56
C500-9_new	550	10	1,217,617	67,323	14	104.47	35.71%
p-hat700-1_new	770	10	1,033,218	36,474	7	0.33	1597.51
p-hat700-3_new	770	10	2,069,033	109,663	14	93.69	276.93%
p-hat700-2_new	770	10	1,549,343	72,990	12	6.11	166.67%
keller5_new	853	10	2,550,888	135,400	13	189.79	150.00%
brock800-4_new	880	10	2,438,428	124,643	11	73.67	336.36%
brock800-3_new	880	10	2,434,282	124,365	11	76.30	336.36%
brock800-2_new	880	10	2,442,137	124,624	12	70.78	336.36%
brock800-1_new	880	10	2,436,213	124,608	11	73.47	336.36%
p-hat1000-3_new	1,100	10	4,209,663	223,153	14	1665.42	400.00%
DSJC1000-5_new	1,100	10	3,174,124	149,879	9	48.18	433.33%
p-hat1000-2_new	1,100	10	3,131,255	146,909	13	37.52	275.00%
p-hat1000-1_new	1,100	10	2,089,333	73,221	7	1.22	257.14%
san1000_new	1,100	10	3,180,173	149,990	9	181.12	66.67%
p-hat1500-1_new	1,650	10	4,785,882	170,681	8	7.27	462.50%

Table 2: Wall-clock running times (in seconds) observed using the EP+WB and EF solvers for the maximum atomic clique problem on SNAP temporal graphs. The entry "LPNS" means that the root LP relaxation was not solved to optimality under the 1-hour time limit. The entry "MEM" indicates that the solver did not terminate gracefully due to a memory-related crash.

						Wall-clock time	
Name	$\lvert V(G^0)\rvert$	$\lvert\mathcal{G}\rvert$	$\sum_{G\in\mathcal{G}}\lvert E(G)\rvert$	$\lvert E(\widehat{G})\rvert$	Objective	EP+WB	EF
CollegeMsg_new	1,899	7	15,714	146	4	0.01	116.46
sx-mathoverflow_new	24,818	8	213,564	1,429	3	0.13	LPNS
sx-askubuntu_new	159,316	8	464,237	23,238	3	0.64	MEM
sx-superuser_new	194,085	9	734,144	19,227	3	0.87	MEM
wiki-talk-temporal_new	1,140,149	8	2,872,615	112,510	5	3.61	MEM
sx-stackoverflow_new	2,601,977	9	28,879,562	220,687	4	37.05	MEM