# A Sparse Matrix Approach for Covering Large Complex Networks by Cliques

W. M. Abdullah[1], S. Hossain[2], M. A. Khan[3]

[1] University of Lethbridge, Alberta, Canada
w.abdullah@uleth.ca
[2] University of Lethbridge, Alberta, Canada
shahadat.hossain@uleth.ca
[3] InBridge Inc, Lethbridge, Alberta, Canada
muhammad@inbridgeinc.com

**Abstract**

A classical NP-hard problem is the Edge Clique Cover (ECC) problem, which is concerned with covering edges of a graph with the minimum number of cliques. There are many real-life applications of this problem, such as, in food science, computational biology, efficient representation of pairwise information and so on. Based on sparse matrix data structures, in this work we propose using a compact representation of network data. We proffer selecting edges by their degree-based orders during the clique-cover step of an existing heuristic. On a set of standard benchmark instances our ordered approach produced smaller sized clique covers compared to random unordered processing in most of the instances.

**Keywords** : *Adjacency matrix, Clique cover, Intersection matrix, Ordering, Sparse graph.*

## 1 Introduction

The graph kernel operations, such as, identification of and computation with dense subgraphs that arise in areas as diverse as sparse matrix determination and complex network analysis [8, 7]. In social networks, identifying special interest groups or characterizing information propagation are examples of frequently performed operations [12]. Effective representation of network data is critical to meeting algorithmic challenges especially for very large and sparse graphs. In this paper, we propose sparse matrix data structures to enable compact representation of graph data and use an existing sparse matrix framework [5] to design efficient algorithms for the ECC problem.

Let $G = (V, E)$ be an undirected connected graph, where $V$ is the set of vertices and $E$ is the set of edges. A clique is a subset of vertices such that every pair of distinct vertices are connected by an edge in the induced subgraph. In graph $G$, an edge clique cover of size $k$ is a decomposition of set $V$ into $k$ subsets $C_1, C_2, \ldots, C_k$ such that $C_i, i = 1, 2, \ldots, k$ induces a clique in $G$ and each edge $\{u, v\} \in E$ is included in some $C_i$. A trivial clique cover can be specified by the set of edges $E$ with each edge being a clique. Finding a clique cover with minimum number of cliques (and many variants thereof) is known to be NP-hard problem[10].

In the literature, many heuristics have been proposed to approximately solve ECC problem while there are only few exact methods which are usually limited to solving small instance sizes. A recent approach is described by Conte et al. in [2], where they introduce $O(m\Delta)$ heuristic to cover all edges of given graph, where $m$ is the number of edges and $\Delta$ is the highest degree of the graph.

In this paper we propose a compact representation of network data based on sparse matrix data structures [6] and provide an improved algorithm based on an existing heuristic [2] for finding clique cover. In [1], we proposed similar compact representation of network data which produced smaller sized clique cover than [4].

Our approach is based on the simple but critical observation that for a sparse matrix $A \in \mathbb{R}^{m \times n}$, the row intersection graph of $A$ is isomorphic to the adjacency graph of $AA^\top$, and that the column intersection graph of $A$ is isomorphic to the adjacency graph of $A^\top A$ [5]. Therefore, the subset of rows corresponding to nonzero entries in column $j$ induces a clique in the adjacency graph of $AA^\top$, and the subset of columns corresponding to nonzero entries in row $i$ induces a clique in the adjacency graph of $A^\top A$. Note that, matrices $A^\top A$ and $AA^\top$ are most

likely dense even if matrix $A$ is sparse. In this work, we exploit the connection between sparse matrices and graphs in the reverse direction. We show that given a graph (or network), we can define a sparse matrix, *intersection matrix*, such that graph algorithms of interest can be expressed in terms of the associated intersection matrix. This structural reduction enables us to use existing sparse matrix computational framework to solve graph problems [5]. This duality between graphs and sparse matrices has also been exploited where the graph algorithms are expressed in the language of sparse linear algebra [8, 9]. However, they use adjacency matrix representation which is different from our intersection matrix representation.

# 2 Compact Representation and Edge Clique Cover

For efficient computer implementation of many important graph operations, representing graphs using adjacency matrix and adjacency list are inadequate. Adjacency matrix is costly for sparse graphs and typical adjacency list implementations employ pointers where indirect access leads to poor cache utilization. The intersection matrix representation that we propose below enables efficient representation of pairwise information and allows us to utilize computational framework DSJM to implement the ECC algorithm.

## 2.1 Intersection Matrix

We require some preliminary definitions. The *adjacency graph* associated with a matrix $A \in \mathbb{R}^{n\times n}$ is a graph $G = (V, E)$ in which for each column or row $k$ of $A$ there is a vertex $v_k \in V$ and $A(i, j) \neq 0$ implies $\{v_i, v_j\} \in E$. The *column intersection graph* associated with matrix $A \in \mathbb{R}^{m\times n}$ is a graph $G = (V, E)$ in which for each column $k$ of $A$ there is a vertex $v_k \in V$ and $\{v_i, v_j\} \in E$ whenever there is a row $l$ for which $A(l, i) \neq 0$ and $A(l, j) \neq 0$.

Let $G = (V, E)$ be an undirected and connected graph without self-loops or multiple edges between a pair of vertices. The adjacency matrix $A(G) \equiv A \in \{0, 1\}^{|V|\times|V|}$ associated with graph $G$ is defined as,

$$A(i, j) = \begin{cases} 1 & \text{if } \{v_i, v_j\} \text{ where } i \neq j \text{ is in } E \\ 0 & \text{otherwise} \end{cases}$$

Let the edges in $E$ be labeled $e_1, \ldots, e_{|E|}$. An intersection matrix associated with graph $G = (V, E)$ where $|V| = n$ and $|E| = m$, is a matrix $C \in \{0, 1\}^{m\times n}$ where for edge $e_k = \{v_i, v_j\}, k = 1, \ldots, m$ we have $C(k, i) = C(k, j) = 1$, and all other entries of matrix $C$ are zero.

Let $C \in \{0, 1\}^{m\times n}$ be the intersection matrix as defined above associated with a graph $G = (V, E)$. Consider the product $B = C^\top C$.

**Theorem 1** *The adjacency graph of matrix $B$ is isomorphic to graph $G$. [1]*

Theorem 1 establishes the desired connection between a graph and its sparse matrix representation. The following result follows directly from Theorem 1.

**Corollary 1** *The diagonal entry $B(i, i)$ where $B = C^\top C$ and $C$ is the intersection matrix of graph $G$, is the degree $d(v_i)$ of vertex $v_i \in V, i = 1, \ldots, n$ of graph $G = (V, E)$. [1]*

Intersection matrix $C$ defined above represents an edge clique cover of cardinality $m$ for graph $G$. Each edge $\{v_i, v_j\}$ constitutes a clique of size 2. In the intersection matrix $C$, edge $e_k = \{v_i, v_j\}$ is represented by row $k$ with $C(k, i) = C(k, j) = 1$ and other entries in the row being zero. In general, column indices $l$ in row $k$ where $C(k, l) = 1$ constitutes a clique on vertices $v_l$ of graph $G$. Thus the ECC problem can be cast as a *matrix compression* problem.

**ECC Matrix Problem.** Given $A \in \{0, 1\}^{m\times n}$ determine $A' \in \{0, 1\}^{k\times n}$ with $k$ minimized such that the intersection graphs of $A$ and $A'$ are isomorphic.

# 3 Clique Cover using a heuristic

The heuristic algorithm that we have implemented for the ECC problem is based on an algorithm due to Conte et al. in [2]. For ease of presentation we discuss the algorithm in graph

theoretic terms. However, our computer implementation uses sparse matrix framework of DSJM [5] and all computations are expressed in terms of intersection matrices.

We employ three vertex ordering algorithms from the literature: Largest-first order (LFO), Smallest-Last Order (SLO), and Incidence-degree Order (IDO) prior to applying the heuristic [2].

For a vertex $v \in V$ we define by $N_v = \{w \in V \mid \{v, w\} \in E\}$ the set of its neighbors. We recall that $d(v) = |N_v|$ denotes the degree of vertex $v$ in graph $G = (V, E)$. **LFO** orders the vertices such that $\{d(v_i), i = 1, \ldots, n\}$ is nonincreasing. **SLO** assumes that the last $n - k$ vertices $\{v_{k+1} \ldots, v_n\}$ in smallest-last order have been determined. The $k^{th}$ vertex in the order is an unordered vertex whose degree in the subgraph induced by $V \setminus \{v_{k+1}, \ldots, v_n\}$ is minimum. **IDO** assumes that the first $k - 1$ vertices $\{v_1 \ldots, v_{k-1}\}$ in incidence-degree order have been determined. Choose $v_k$ from among the unordered vertices that has maximum degree in the subgraph induced by $\{v_1, \ldots, v_k\}$.

Next, we present a new algorithm for the ECC problem.

Let the vertices of graph $G = (V, E)$ be ordered in one of SLO, LFO, and IDO: $v_1, \ldots, v_n$. Then we order edges considering the vertex order, such that, $e_i \in E$ will be considered before $e_j \in E$ if one of the vertices of $e_i$ is ordered before the vertices of $e_j$.

Also, let $E_{\mathcal{P}} = \{e_1, \ldots, e_{i-1}\}$ denote the edges that have been assigned to one or more cliques $\{C_1, \ldots, C_{k-1}\}$ and $e_i = \{v_i, v_j\}$ be the edge currently being processed. Denote by set

$$W = \{v_l \mid \{v_i, v_l\}, \{v_j, v_l\} \in E\}$$

the common neighbors of $v_i$ and $v_j$. The task is to assign edge $\{v_i, v_j\}$ (if not covered yet) to one new clique and add its common neighbors if they satisfy clique properties.

The complete algorithm is presented below.

**CliqueDecomp** ($W$, *list*)

```
1:  k ← 0                                                  ▷ Number of cliques
2:  for index = 1 to m do                                  ▷ m is number of edges
3:      {u, v} ← list[index]                               ▷ list contains the edges in a predefined order
4:      if {u, v} is not covered then
5:          W ← FindCommonNeighbours(u, v)
6:          if W = ∅ then
7:              k++
8:              C_k ← {u, v}
9:              Mark {u, v} as covered.
10:         else
11:             k++
12:             C_k ← {u, v}
13:             Mark {u, v} as covered.
14:             while W ≠ ∅ do
15:                 t ← take a vertex from W
16:                 if t has edges with all s ∈ C_k then
17:                     Mark {t, s} as covered.
18:                     C_k ← C_k ∪ {t}
19:                     W = W ∩ Neighbor_t                 ▷ Neighbor_t denotes the neighbors of vertex t
20: return C_1, C_2, ..., C_k
```

### 3.1 Discussion

Line 5 calls $FindCommonNeighbours$ and Line 19 finds $Neighbor_t$. We have implemented these using DSJM [5] which takes liner time for these operations. Because, in our implementation to find the neighbors for each non-zero entries of the selected column we look for the non-zero entries in the same row. So in total each row is searched $\rho_i^2$ times where $\rho_i$ is the number of nonzero elements in the row $i$. If we have $m$ rows (outer loop runs for $m$ times in line 2), sequential ordering will take $O(\sum_{i=1}^{m} \rho_i^2)$ operations. Line 16 takes $O(\Delta)$ times, because we look for $\Delta$ number of edges for vertex $t$. Hence this heuristic takes $O(m\Delta)$ time. Where $m$ is the number of edges and $\Delta$ denotes the degree of a vertex.

# 4 Numerical Testing

In this section, we provide results from numerical experiments on selected test instances. The data set for the experiments is obtained from the University of Florida Sparse Matrix Collection [3]. The experiments were performed using a PC with 3.4G Hz Intel Xeon CPU, 8 GB RAM running Linux. The implementation language was *C++* and the code was compiled using $-O2$ optimization flag with a *g++* version 4.4.7 compiler.

Stanford Network Analysis Platform (`SNAP`) is a collection of more than 50 large network datasets containing large number of nodes and edges including social networks, web graphs, road networks, internet networks, citation networks, collaboration networks, and communication networks [11].

Test results for the selected test instances from group `SNAP` are reported in Table 1. Here, $n$ represents the number of vertices and $m$ represents the number of edges of the graph. $|C|$ represents number of cliques required to cover all the edges.

TAB. 1: Test Results (Number of cliques) for SNAP matrices

| Matrix | | | Heuristic-1 | Heuristic-2 | Heuristic-3 |
|---|---|---|---|---|---|
| Name | $m$ | $n$ | (using [1]) | (using [2]) | (Proposed) |
| p2p-Gnutella04 | 39994 | 10878 | 38474 | 38491 | **38449** |
| p2p-Gnutella24 | 65369 | 26518 | 63726 | 63725 | **63689** |
| p2p-Gnutella25 | 54705 | 22687 | 53368 | 53367 | **53347** |
| p2p-Gnutella30 | 88328 | 36682 | 85823 | 85822 | **85717** |
| ca-GrQc | 14496 | 5242 | 3777 | 3753 | **3717** |
| as-735 | 13895 | 7716 | 8985 | **8938** | 10130 |
| Wiki-Vote | 103689 | 8297 | 42914 | **39393** | 51145 |
| Oregon-1 | 23409 | 11492 | 15631 | **15491** | 15527 |
| ca-HepTh | 25998 | 9877 | 9663 | 9270 | **9162** |

Table 1 displays results using our new algorithm (`Heuristic-3`), our old algorithm (`Heuristic-1`) discussed in [1] and algorithm (`Heuristic-2`) discussed in [2]. `Heuristic-1` constructs a clique cover by trying to add the next vertex to an existing clique, whenever possible. On the other hand, `Heuristic-2` randomly selects an edge and attempts to build a clique around the selected edge. `Heuristic-3` produces smaller cardinality ECC than `Heuristic-1` except for two instances. On the other hand, it compares favourably with `Heuristic-2` (six out of nine instances

# 5 Conclusion

In this work we have proposed a compact representation of network data. The edge clique cover problem is recast as a sparse matrix determination problem. The notion of *intersection matrix* provides a unified framework that facilitates compact representation of graph data and efficient implementation of graph algorithms. The adjacency matrix representation of a graph can potentially have many nonzero entries since it is the product of an intersection matrix with its transpose. We have shown that similar to graph vertex coloring problem ECC problem is sensitive to ordering of the vertices and hence ordering of the edges.

# References


[1] W. M. Abdullah, S. Hossain, and M. A. Khan. Covering Large Complex Networks by Cliques - A Sparse Matrix Approach. *AMMCS 2019 International Conference, Waterloo, Canada*, 2019.

[2] Alessio Conte, Roberto Grossi, and Andrea Marino. Large-scale clique cover of real-world networks. *Information and Computation*, 270:104464, 2020.

[3] T. Davis and Y. Hu. Suitesparse matrix collection. https://sparse.tamu.edu/. Accessed: 2019-10-02.

[4] J. Gramm, J. Guo, F. Huffner, and R. Niedermeier. Data reduction, Exact and Heuristic Algorithms for Clique Cover. *Proceedings of the Eighth Workshop on Algorithm Engineering and Experiments (ALENEX), SIAM*, pages 86–94, 2006.

[5] M. Hasan, S. Hossain, A. I. Khan, N. H. Mithila, and A. H. Suny. DSJM: A Software Toolkit for Direct Determination of Sparse Jacobian Matrices. In: G.M. Greuel, T. Koch,

P. Paule, A. Sommese and Editors. ICMS2016. *Springer International Publishing Switzerland*, pages 425–434, 2016.

[6] S. Hossain and A. I. Khan. Exact Coloring of Sparse Matrices. In: D.M. Kilgour et al. (eds.) Recent Advances in Mathematical and Statistical Methods. *Springer Proceedings in Mathematics and Statistics, Springer Nature Switzerland AG*, 259:23–36, 2018.

[7] S. Hossain and A. H. Suny. Determination of Large Sparse Derivative Matrices: Structural: Orthogonality and Structural Degeneracy. *In: B. Randerath, H. Roglin, B. Peis, O. Schaudt, R. Schrader, F. Vallentin and V. Weil. 15th Cologne-Twente Workshop on Graphs & Combinatorial Optimization, Cologne, Germany*, pages 83–87, 2017.

[8] J. Kepner and J. Gilbert. Graph Algorithms in the Language of Linear Algebra, Society for Industrial and Applied Mathematics. *Philadelphia, PA, USA*, 2011.

[9] J. Kepner and H. Jananthan. Mathematics of big data: Spreadsheets, databases, matrices, and graphs. *MIT Press*, 2018.

[10] LT Kou, LJ Stockmeyer, and CK Wong. Covering edges by cliques with regard to keyword conflicts and intersection graphs. *Communications of the ACM*, 21(2):135–139, 1978.

[11] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014. Accessed: 2019-10-02.

[12] S Wasserman and K Faust. Social network analysis: Methods and applications. *Cambridge university press*, 1994.