Flexi-clique: Exploring Flexible and Sub-linear Clique Structures

Song Kim
Ulsan National Institute of Science & Technology
Ulsan, South Korea
song.kim@unist.ac.kr

Junghoon Kim*
Ulsan National Institute of Science & Technology
Ulsan, South Korea
junghoon.kim@unist.ac.kr

Susik Yoon
Korea University
Seoul, South Korea
susik@korea.ac.kr

Jungeun Kim
Inha University
Incheon, South Korea
jekim@inha.ac.kr

Abstract

Identifying cohesive subgraphs within networks is a fundamental problem in graph theory, relevant to various domains. The traditional clique problem, which finds fully connected subgraphs, often faces limitations due to its strict connectivity requirements. This paper introduces a novel degree-based relaxation model called Flexi−clique, where the degree constraint is adjusted sub-linearly based on the subgraph size. We establish that the maximum Flexi−clique problem is NP-hard and propose an efficient and effective peeling algorithm to address it. Our extensive experimental evaluation of real-world datasets demonstrates the effectiveness and efficiency of our approach in discovering large, cohesive subgraphs in networks.

1 Introduction

How can we identify the structure with the high cohesiveness in a network? The clique problem [17] provides an answer to this question. Given a simple and undirected graph, a clique is defined as a set of nodes within a graph, every two of which are connected. The unique characteristic of cliques serves as an effective and powerful tool to find tightly-knit subgraphs within various networks. Thus, it has gained significant interest and has been utilised in numerous applications such as graph clustering [20], economics [6], coding theory [12], scheduling [11], and bioinformatics [10].

While the structure of a clique has practical implications, the criteria for identifying one are quite strict, requiring connections between every pair of nodes. In practice, therefore, it is known that the size of the maximum clique is relatively small even in very large graphs [7, 15]. In sociology, it is understood that as the size of a community grows, the connectivity between individuals tends to weaken, but the influence of a community becomes more pronounced [23]. This concept leads to active discussions about exploring relaxed clique structures, especially with large subgraphs.

Several models mitigate clique constraints. The s-clique [16] allows nodes within distance s as neighbours. The s-club [19] focuses on the maximum distance between nodes within the subgraph. The quasi-clique [1] specifies a threshold for minimum density. The k-plex [22] allows a specified number of missing connections, useful for real-world networks where perfect cliques are rare.

While the k-plex has proven useful, its degree constraint increases linearly with the size of the subgraph which fundamentally limits the flexibility of approximation over varying sizes of the graph. For smaller graphs, allowing k disconnections within a subgraph can make the constraints overly relaxed. For larger graphs, on the other hand, it results in a similar strictness as cliques, making it difficult to find large k-plexes [18] when the subgraph size becomes very large.

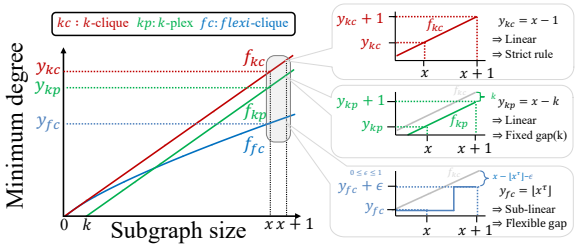


Figure 1: Motivation figure

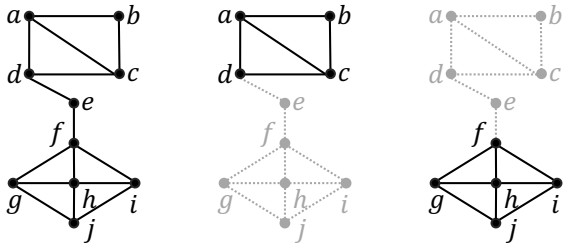
To address the issue of k-plex, we propose Flexi−clique, a new degree-based relaxation model where the degree constraint is bounded by a given exponent regarding the subgraph size. This approach offers two main benefits: (1) *flexibility*: by allowing the degree constraint to adapt to the size of the subgraph, our model can capture meaningful subgraphs in networks with varying densities, and (2) *sub-linearity*: the minimum degree increases less strictly than linearly with the graph size, making it feasible to find larger cohesive subgraphs even in very huge networks.

Figure 1 shows the differences between the k-clique, k-plex, and the Flexi−clique. It demonstrates that k-clique, which is a clique with a size of k, demands a strict minimum degree proportional to the subgraph size. For k-plex, two aspects can be observed. Firstly, it does not produce meaningful result until the subgraph size reaches k, meaning that when k is large, it may ignore small subgraphs having dense structures. On the other hand, as the subgraph size increases, the minimum degree of k-plex also increases linearly. This implies that the strictness issue in k-clique could arise in the k-plex model as well when the subgraph size becomes bigger. However, Flexi−clique has strict connectivity when the subgraph size is small but gradually relaxes the constraints as the subgraph size increases. This sub-linear adjustment allows the model to adapt more flexibly to different subgraph sizes, maintaining a balance between connectivity and flexibility. Consequently, Flexi−clique can effectively identify larger, cohesive structures that other models might have missed.

In this paper, we study the maximum Flexi−clique problem. Given a graph and a user-defined parameter, a Flexi−clique is a connected subgraph where each node degree satisfies a sub-linear threshold relative to the subgraph size. The goal is to identify the largest subgraph satisfying these conditions. This problem is challenging due to the combinatorial nature of subgraph discovery and the dynamic adjustment of the degree constraint. We prove that the problem is NP-hard, propose an efficient peeling algorithm, and conduct extensive experiments on real-world networks to demonstrate the superiority of our proposed algorithm.

*Corresponding author

Conference'17, Washington, DC, USA
2025. ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn



(a) Input graph G (b) Flexi−clique (c) max. Flexi−clique

Figure 2: Flexi−clique example($\tau = 0.7$)

Contributions. Our contributions are summarised as follows: (1) Novel relaxation model: To the best of our knowledge, this work introduces the first sub-linear degree-based clique relaxation for the efficient discovery of cohesive subgraphs; (2) Proving NP-hardness: We theoretically prove the NP-hardness of the maximum Flexi−clique problem; and (3) Extensive experiments: With real-world network datasets, we have conducted a comprehensive experimental study, demonstrating the efficiency of our peeling algorithms.

2 Related Work

Cohesive subgraph discovery is a fundamental problem in graph mining, with a wide range of applications in social network analysis, bioinformatics, and recommendation systems. Over the years, several models have been proposed to define and extract dense subgraphs, including cliques, quasi-cliques, and k-plexes. These models offer varying degrees of density and connectivity constraints, making them suitable for different use cases.

Clique is one of the earliest and most strict definitions of cohesive subgraphs. A clique is a subset of vertices in which every pair of vertices is connected by an edge. Given its strict requirement of complete connectivity, the maximum clique problem is known to be NP-hard [13]. While cliques capture the most tightly connected structures in a graph, their strict definition often limits their applicability to small subgraphs in real-world networks.

To address this limitation, quasi-cliques [1] were introduced as a relaxation of cliques, allowing some missing edges while still maintaining a high level of connectivity. Specifically, a quasi-clique ensures that a large proportion of possible edges exist within the subgraph. This relaxation increases the size of the subgraphs that can be discovered, making quasi-cliques useful in scenarios where perfect connectivity is not required.

Another popular relaxation model is the k-plex, which further generalises cliques by allowing each vertex to miss up to k connections within the subgraph [22]. Formally, a k-plex of size n is a subgraph where each vertex is adjacent to at least $n - k$ other vertices. This definition allows for significant flexibility, making k-plexes particularly useful for handling noisy or incomplete graphs, such as those encountered in biological networks.

Although various heuristic and exact algorithms have been proposed to discover cohesive subgraphs efficiently, existing models often struggle when applied to large networks. Their strict or unrealistic connectivity constraints can fail to capture flexible and meaningful substructures in real-world graphs. To address these challenges, we propose a novel model that balances subgraph size, density, and computational efficiency.

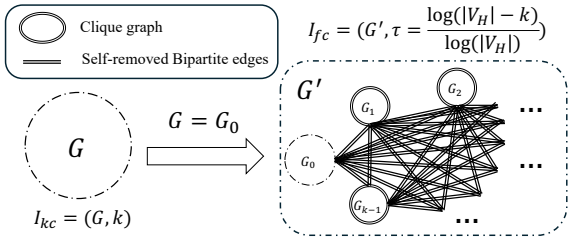


Figure 3: Reduction process

3 Problem Statement

In this work, let $G = (V, E)$ be a simple undirected graph, $G[S]$ the subgraph induced by $S \subseteq V$, $d(u, G)$ the degree of node u in G, and $\delta(G)$ the minimum degree in G. We first formally define Flexi−clique, then formulate the maximum Flexi−clique problem.

Definition 1. *(Flexi−clique.) Given a graph $G = (V, E)$ and a parameter τ, a Flexi−clique, denoted as H, is a node set such that:*

- *Every node $v \in H$ has at least $\lfloor |H|^{\tau} \rfloor$ degree in $G[H]$.*
- *$G[H]$ is connected.*

Problem definition 1. *(Maximum Flexi−clique problem.) Given a graph $G = (V, E)$ and a parameter τ, the maximum Flexi−clique problem is to find the largest Flexi−clique of the given graph.*

Example 1. *Given a graph G in Figure 2a with $\tau = 0.7$, the set of nodes a, b, c, d forms a Flexi−clique since it is connected and each node has at least $\lfloor 4^{0.7} \rfloor = 2$ neighbours. Similarly, the set of nodes f, g, h, i, j also forms a Flexi−clique as it satisfies the connectivity and degree constraints. This example shows that multiple Flexi−cliques can exist within the same graph.*

The maximum Flexi−clique problem is challenging due to the combinatorial nature of subgraph discovery and the dynamic constraint on the minimum degree depending on the subgraph size. We now formally establish the computational hardness of this problem.

Theorem 1. *Maximum Flexi−clique problem is NP-hard.*

Proof. To prove the NP-hardness of the maximum Flexi−clique problem, we employ a novel reduction technique from the k-clique to k-plex [3] by incorporating an additional threshold τ. Given a graph $G = (V, E)$ and an integer k, the decision version of the k-clique problem is to determine if there is a clique of size k. We denote an instance of the k-clique problem as $I_{kc} = (G, k)$. From the instance I_{kc}, we form an instance $I_{fc} = (G', \tau)$. In this instance, we aim to claim that identifying the maximum Flexi−clique is at least as hard as finding a k-clique in I_{kc}. The graph generation and setting of the parameter τ are processed as follows.

Formation from the instance I_{kc} to I_{fc} is presented in Figure 3. First, we consider $G_0 = (V, E)$, which is the same as G of I_{kc} and we denote n as $|V|$. Our newly generated graph G' consists of $G_0, G_1, \ldots, G_{k-1}$. Note that for all G_i where $1 \leq i \leq k - 1$, each graph G_i contains n nodes and forms a clique, i.e., there are $\frac{n(n-1)}{2}$ edges in G_i. We call them clique graphs. In addition, any pair of graphs G_x and G_y where $x \neq y$ and $0 \leq x, y \leq k - 1$ are connected by self-removed bipartite edges. The self-removed bipartite edges form a set of all edges from node $u \in V_x$ to $v \in V_y$ where $u \neq v$. Thus, each node $u \in V_x$ has $n - 1$ connections to V_y.

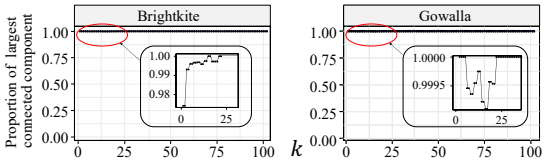


Figure 4: Proportion of the largest k-core component

Next, we set τ as $\frac{\log(|H|-k)}{\log(|H|)}$ where $|H|$ is a size of nodes of a connected subgraph $G[H]$:

$$\lfloor |H|^{\frac{\log(|H|-k)}{\log(|H|)}} \rfloor \Rightarrow \lfloor |H|^{\log_{|H|}(|H|-k)} \rfloor \Rightarrow |H| - k \tag{1}$$

Given an instance $I_{fc} = (G', \tau)$, we claim that the maximum Flexi−clique is the set of nodes $H = C \cup V_1 \cup V_2 \cup \cdots \cup V_{k-1}$ where C is a set of nodes in a clique of size k in G_0. The size of H is $n(k-1)+k$. Details in the degree of the nodes in H are as follows:

- Each node $v \in C$ has $k-1$ neighbours in $G_0[C]$ and $(n-1)(k-1)$ neighbours in clique graphs $G_1 \cup \cdots \cup G_{k-1}$.
- Each node in clique graphs has at least $k-1$ neighbours in $G_0[C]$, $n-1$ neighbours in the same clique graph, and $(n-1)(k-2)$ neighbours in other clique graphs.

From this, we can find the relationship between the minimum degree of the nodes in H and the size of H:

$$\begin{aligned} \delta(H) &= k - 1 + (n-1)(k-1) = n(k-1) \\ &= n(k-1) + k - k = |H| - k \end{aligned} \tag{2}$$

Therefore, it indicates that the identified nodes in H satisfy the constraint of Flexi−clique. This directly indicates that identifying Flexi−clique in I_{fc} is at least as hard as finding a k-clique in I_{kc}. Thus, Flexi−clique is NP-hard. □

4 Algorithm

We present a non-articulation peeling algorithm (NPA) to find the maximum Flexi−clique. The high-level idea is to iteratively remove nodes while maintaining connectivity. For large graphs, the initial graph may require a high minimum degree. However, in real-world networks [4, 9], many nodes have low degrees. Thus, we utilise k-core [21], a set of nodes where each has at least k neighbours, to prune the nodes. The k-core can have multiple components, but we focus on the largest. Our empirical study shows that k-core returns few components, with the largest containing most nodes (Figure 4).

Algorithm description. The pseudo description for the algorithm is presented in Algorithm 1. The NPA increases k' and finds all connected components for the k'-core. It stores all components, except the largest one, in a priority queue along with the current k' value. If the largest connected component is a Flexi−clique, it is added to the queue; otherwise, it is inserted into the priority queue (Lines 4-10). After traversing all possible k-cores, it iteratively retrieves subgraphs from the queue to build the initial graph for the peeling process. For each graph, non-articulation node with the minimum degree is iteratively removed while preserving the connectivity. If a Flexi−clique is successfully found, the set of nodes in the subgraph is returned as the result (Lines 11-21). Otherwise, the next subgraph from the queue is processed. If the queue is empty, a subgraph is picked from the priority queue. This process is repeated until we identify a Flexi−clique. Note that this approach

Algorithm 1: Peeling algorithm for the Flexi−clique (NPA)

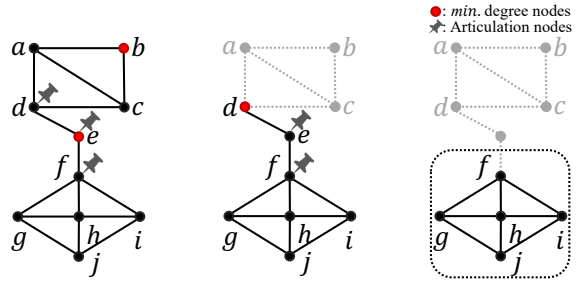
Input : A graph $G = (V, E)$ and a threshold parameter τ
Output: A Flexi−clique H

```
1  c ← coreness(G) // Map storing coreness
2  PQ ← PriorityQueue() // Sort by size
3  Q ← Queue(), T ← ∅, k* ← maxValue(c);
4  for (k' ← 2; k' ≤ k*; k'++) do
5      D ← kcore(k', c);
6      C ← connectedComps(D);
7      T ← arg max_{t∈C} |t|;
8      PQ.add((t, k')), ∀t ∈ C \ T;
9      if ⌊|T|^τ⌋ ≤ k' then Q.add((T, k')) ;
10     else PQ.add((T, k')) ;
11 while Q ≠ ∅ OR PQ ≠ ∅ do
12     if Q ≠ ∅ then T, k' ← Q.pop() ;
13     else T, k' ← PQ.pop() ;
14     k' ← k' − 1;
15     H ← kcoreCC(k', T, c);
16     while δ(H) < ⌊|H|^τ⌋ do
17         A ← articulationNodes(G[H]);
18         T ← H \ A;
19         u ← arg min_{v∈T} d(v, G[H]);
20         H ← H \ {u};
21     if H ≠ ∅ then return H ;
22 return ∅;
```

puts a high priority on the largest connected component of each k-core to align with our objective function.

Time complexity. $O(k^*|V| \cdot (|V|+|E|))$ is the total time complexity of the NPA. The complexities of the key components are as follows:

- Coreness: Computing coreness of the nodes takes $O(|E|)$ time [5].
- Finding connected components: Retrieving connected components takes $O(|V|+|E|)$ time, and this needs to be done k^* times.
- Articulation nodes identification: This step can be performed in $O(|V|+|E|)$ time using a DFS-tree.
- Number of iteration: Given a specific subgraph, the number of iterations for the removal process is $O(|V|)$.



(a) Input graph G **(b) Peeling process** **(c) Result**

Figure 5: NPA example

Example 2. *Using the graph in Figure 2 with $\tau = 0.7$, we present the procedure of NPA in Figure 5. Nodes with the minimum degree are marked in red and articulation nodes are marked as "pinned". The algorithm first finds the largest k-core component satisfying the Flexi−clique constraint. Since the set $\{f, g, h, i, j\}$ forms both a 3-core and a Flexi−clique, the peeling process starts from the 2-core*

Table 1: Real-world datasets

Dataset	No. of nodes	No. of edges	No. of comm.
Karate (KT)	34	78	2
Brightkite (BK)	58,228	214,078	-
Gowalla (GW)	196,591	950,327	-
Amazon (AZ)	334,863	925,827	75,149

containing the set, which is the whole graph. The algorithm iteratively checks and removes nodes not meeting the degree constraint while preserving connectivity. Thus, in Figure 5a, node e is not removed despite having the minimum degree. Figures 5b and 5c illustrate changes in articulation points during the process. Ultimately, the set $\{f, g, h, i, j\}$ *is identified as the resultant Flexi−clique.*

5 Experiments

5.1 Experimental setup.

To evaluate the performance of our model and algorithm, we conduct an extensive experimental study. Through our experiments, we aim to address the following evaluation questions (**EQs**): (1) **Effectiveness**: How accurately does our algorithm perform compared to the exact solution?; (2) **Effect on** τ: How does varying τ impact resultant subgraphs? We analyse the effects of different τ values on subgraph size; (3) **Scalability**: How well does the proposed algorithm scale on datasets with different sizes?; and (4) **Effect on** k**-core.** How much does using k-core as a preprocessing step improve the running time of the peeling algorithm?

5.2 Experimental setting

Real-world datasets. Table 1 provides the basic information of real-world datasets used in our experiments. These datasets are publicly available, as referenced [8, 24]. Note that Gowalla and Brightkite datasets have no ground-truth community information.

5.3 Experimental result

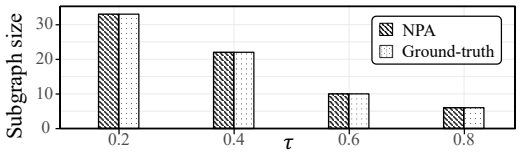


Figure 6: EQ1. Comparing with Ground-truth

EQ1. Effectiveness. In this experiment, we compare the results of NPA with the optimum solution obtained by enumerating all connected components in the small network(KT) [25] using publicly available code [2]. Surprisingly, Figure 6 shows that the proposed NPA returned the same results as the optimum solution for all four different τ values tested. This remarkable consistency demonstrates the experimental effectiveness and robustness of our algorithm.

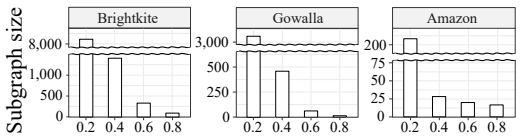


Figure 7: EQ2. Effect on τ

EQ2. Effect on τ**.** We analyse how varying the τ of NPA impacts the subgraph size. Figure 7 presents the results. As τ increases, the size of the Flexi−clique consistently decreases because a larger τ indicates a search for a more cohesive subgraph structure. This increased cohesiveness requirement means that fewer nodes can satisfy the stricter connectivity, resulting in smaller subgraphs.

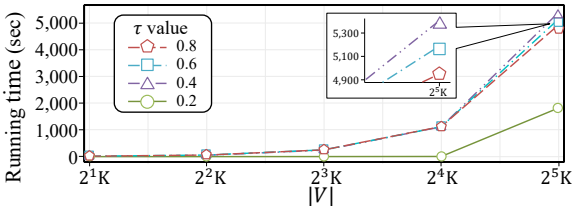


Figure 8: EQ3. Scalability test

EQ3. Scalability test. We evaluate the scalability of our proposed algorithm on synthetic datasets [14]. Using default parameters, we generate graphs varying in size from 2,000 to 32,000 nodes under different τ conditions. Figure 8 shows a near-linear increase in running time as the node size grows. Additionally, we observe that as τ increases, the running time does not consistently rise. This result can be explained as follows: For very small τ values, the nodes in the initial graph mostly satisfy the degree constraint early in the peeling process, requiring minimal iterations and resulting in a short running time. Conversely, for larger τ values, most nodes might be addressed during the preprocessing step which also leads to relatively short running time. Therefore, the algorithm takes the longest time for τ values that are neither too small nor too large.

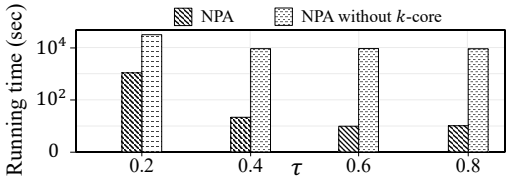


Figure 9: EQ4. Effect on k**-core (Brightkite)**

EQ4. Effect on k**-core.** In our algorithm, we use the k-core connected component as a first step to find the initial subgraph. We observed the results with and without k-core preprocessing. Although the resultant nodes and their sizes were the same in various τ conditions, as shown in Figure 9, the running time differed significantly in the Brightkite dataset. This discrepancy is due to the k-core preprocessing significantly reducing the size of the initial graph, which decreases the time required to compute non-articulation nodes, a process with a time complexity of $O(|V| + |E|)$.

6 Conclusion

In this paper, we introduced the Flexi−clique problem, a more flexible and generalised classical clique problem, and proposed a non-articulation peeling algorithm (NPA) to efficiently identify Flexi−clique subgraphs in large networks. Our theoretical analysis established the NP-hardness of the maximum Flexi−clique problem. Through extensive experiments, we demonstrated the efficiency of our algorithm on both synthetic and real-world datasets. Future work will focus on designing more effective approximation algorithms and considering index-based approach to efficiently retrieve Flexi−clique in dynamic networks.

References

[1] James Abello, Mauricio GC Resende, and Sandra Sudarsky. 2002. Massive quasi-clique detection. In *LATIN 2002: Theoretical Informatics: 5th Latin American Symposium Cancun, Mexico, April 3–6, 2002 Proceedings 5*. Springer, 598–612.

[2] Mohammed Alokshiya, Saeed Salem, and Fidaa Abed. 2018. A linear delay linear space algorithm for enumeration of all connected induced subgraphs. In *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*. 607–607.

[3] Balabhaskar Balasundaram, Sergiy Butenko, and Illya V Hicks. 2011. Clique relaxations in social network analysis: The maximum k-plex problem. *Operations Research* 59, 1 (2011), 133–142.

[4] Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *science* 286, 5439 (1999), 509–512.

[5] Vladimir Batagelj and Matjaz Zaversnik. 2003. An o (m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049* (2003).

[6] Vladimir Boginski, Sergiy Butenko, and Panos M Pardalos. 2006. Mining market data: A network approach. *Computers & Operations Research* 33, 11 (2006), 3171–3184.

[7] Lijun Chang. 2019. Efficient maximum clique computation over large sparse graphs. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 529–538.

[8] Eunjoon Cho, Seth A Myers, and Jure Leskovec. 2011. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1082–1090.

[9] Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. 2009. Power-law distributions in empirical data. *SIAM review* 51, 4 (2009), 661–703.

[10] William HE Day and David Sankoff. 1986. Computational complexity of inferring phylogenies by compatibility. *Systematic Biology* 35, 2 (1986), 224–229.

[11] Ulrich Dorndorf, Florian Jaehn, and Erwin Pesch. 2008. Modelling robust flight-gate scheduling as a clique partitioning problem. *Transportation Science* 42, 3 (2008), 292–301.

[12] Tuvi Etzion and Patric RJ Ostergard. 1998. Greedy and heuristic algorithms for codes and colorings. *IEEE Transactions on Information Theory* 44, 1 (1998), 382–388.

[13] Richard M Karp. 2010. *Reducibility among combinatorial problems*. Springer.

[14] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. 2008. Benchmark graphs for testing community detection algorithms. *Physical review E* 78, 4 (2008), 046110.

[15] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. 2020. *Mining of massive data sets*. Cambridge university press.

[16] R Duncan Luce. 1950. Connectivity and generalized cliques in sociometric group structure. *Psychometrika* 15, 2 (1950), 169–190.

[17] R Duncan Luce and Albert D Perry. 1949. A method of matrix analysis of group structure. *Psychometrika* 14, 2 (1949), 95–116.

[18] Zhuqi Miao and Balabhaskar Balasundaram. 2017. Approaches for finding cohesive subgroups in large-scale social networks via maximum k-plex detection. *Networks* 69, 4 (2017), 388–407.

[19] Robert J Mokken et al. 1979. Cliques, clubs and clans. *Quality & Quantity* 13, 2 (1979), 161–173.

[20] Satu Elisa Schaeffer. 2007. Graph clustering. *Computer science review* 1, 1 (2007), 27–64.

[21] Stephen B Seidman. 1983. Network structure and minimum degree. *Social networks* 5, 3 (1983), 269–287.

[22] Stephen B Seidman and Brian L Foster. 1978. A graph-theoretic generalization of the clique concept. *Journal of Mathematical sociology* 6, 1 (1978), 139–154.

[23] Georg Simmel. 1950. *The sociology of georg simmel*. Vol. 92892. Simon and Schuster.

[24] Jaewon Yang and Jure Leskovec. 2012. Defining and evaluating network communities based on ground-truth. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*. 1–8.

[25] Wayne W Zachary. 1977. An information flow model for conflict and fission in small groups. *Journal of anthropological research* 33, 4 (1977), 452–473.