ISSN (e): 2250 – 3005 || Volume, 14 || Issue, 5|| Sep. - Oct. – 2024 ||
International Journal of Computational Engineering Research (IJCER)

An Algorithm to Determine the Clique Number of a Split Graph

O.Kettani
Scientific Institute - Mohammed V University in Rabat, Morocco

Abstract
Determining the clique number of a graph, which is the size of the largest complete subgraph, is a fundamental problem in graph theory with applications in network analysis, optimization, and data mining. This paper presents an efficient algorithm specifically designed to determine the clique number of a split graph, a special class of graphs characterized by a vertex set that can be partitioned into a clique and an independent set. The proposed algorithm leverages the structural properties of split graphs to identify the maximum clique in $O(n^4)$ time, outperforming general-purpose algorithms that do not exploit these properties. The algorithm is based on a novel approach that systematically examines the interplay between the clique and independent set, ensuring all possible cliques are evaluated efficiently. This work contributes a significant improvement in the computational aspects of clique number determination, specifically tailored to split graphs, offering both theoretical insights and practical benefits.

Keywords: *clique number of a graph, split graph, algorithm.*

Date of Submission: 03-09-2024 Date of acceptance: 15-09-2024

I. Introduction

The clique number of a graph, defined as the size of the largest complete subgraph (or clique) within it, is a critical parameter in graph theory with numerous applications across various domains, including social network analysis, computational biology, and operations research. The problem of determining the clique number is generally NP-hard for arbitrary graphs, posing significant computational challenges, particularly as the size of the graph grows. However, for specific graph classes, such as split graphs, more efficient solutions can be developed.

Split graphs are a special class of graphs that can be decomposed into two disjoint sets: a clique (a complete subgraph) and an independent set (a set of vertices with no edges between them). This unique structure simplifies certain computational problems, including the determination of the clique number. Despite this simplification, directly applying general algorithms designed for arbitrary graphs often leads to suboptimal performance, missing the opportunity to exploit the structural properties inherent in split graphs.

Existing approaches to finding the clique number typically involve exhaustive search methods or adaptations of general-purpose algorithms that do not fully utilize the specific characteristics of split graphs. While these methods can be applied, they are not optimized for the split graph's structure and often result in increased computational overhead. Therefore, there is a need for specialized algorithms that can efficiently and accurately determine the clique number in split graphs by leveraging their unique properties.

In this paper, we propose a novel algorithm specifically designed to determine the clique number of a split graph. The algorithm takes advantage of the bipartition of split graphs into a clique and an independent set, systematically identifying the maximum clique by evaluating potential subgraphs formed within these partitions. By focusing on the relationships between vertices in the clique and independent set, our algorithm achieves a significant reduction in computational complexity compared to general methods.

We demonstrate the efficiency of our approach through theoretical analysis, showing that it operates in $O(n^4)$ time relative to the number n of vertices in the graph. Our results highlight the advantages of tailoring algorithms to specific graph classes, emphasizing the importance of structural exploitation in graph analysis.

The remainder of this paper is organized as follows: Section 2 reviews related work and outlines the limitations of existing methods. Section 3 details the proposed algorithm and its theoretical basis. Finally, Section 4 discusses the implications of our findings and potential areas for future research.

II. Related Work

Determining the clique number of a graph is a well-studied problem in graph theory, with numerous algorithms proposed for various classes of graphs. For general graphs, the problem is known to be NP-hard, and as such, most research has focused on either approximation algorithms or exact algorithms that are efficient for specific graph classes, including chordal graphs, interval graphs, and split graphs. This section reviews the key related work and situates our contribution within the broader context of existing approaches.

One of the foundational methods for finding the clique number in arbitrary graphs involves backtracking and branch-and-bound techniques, which systematically explore all possible subgraphs to identify the largest clique. While these methods guarantee finding the maximum clique, their computational cost grows exponentially with the size of the graph, making them impractical for large instances. Optimization techniques such as pruning strategies and heuristic-guided searches have been employed to improve efficiency; however, these are still not optimal for split graphs as they do not leverage the structural properties unique to this graph class.

Research on specialized algorithms for specific graph classes has demonstrated that exploiting graph structure can lead to significant efficiency gains. For example, chordal graphs, which contain no induced cycles longer than three vertices, allow for polynomial-time solutions to the clique number problem due to their perfect elimination ordering. Similar structural approaches have been developed for interval graphs and other well-structured graph classes, highlighting the value of tailoring algorithms to specific graph characteristics.

Split graphs, first introduced by Földes and Hammer in the 1970s [1], are distinguished by their ability to be partitioned into a clique and an independent set. Despite this distinctive feature, much of the research on split graphs has focused on recognition problems, graph coloring, and vertex cover solutions, with less emphasis on algorithms specifically designed to determine the clique number. The most straightforward approach for split graphs involves directly counting the number of vertices in the clique partition, but this method fails to account for potential interactions with vertices in the independent set, which can form larger cliques in some configurations.

Recent advances have included heuristic and approximation methods for estimating the clique number in split graphs, often using combinatorial optimization techniques. While these methods improve upon the general-purpose algorithms, they still fall short of achieving the efficiency possible when directly leveraging the split graph's structure. Other approaches have explored dynamic programming and greedy algorithms; however, these methods typically focus on related problems such as maximum independent sets and do not address the clique number directly.

Földes, S., & Hammer, P. L [2] discusses heuristics for maximum clique problems on specific graph classes, including split graphs, and presents methods that can be adapted to estimate clique numbers efficiently.

Boros, E., Golumbic, M. C., & Levit, V. E. (2002) [3] work explores properties of split graphs and introduces combinatorial methods that can be used to approximate clique sizes, focusing on vertex participation in cliques.

Kojima, R., & Yamashita, T. (2018) [4] paper presents approximation algorithms specifically tailored for split graphs, offering insights into their performance relative to exact methods and highlighting the efficiency of heuristic approaches.

Papadopoulos, C., & Tzimas, A. (2017) [5] introduce a heuristic-based approach for finding large cliques in split graphs, emphasizing the trade-offs between accuracy and computational complexity.

Our work diverges from these traditional approaches by introducing an algorithm specifically tailored to the split graph structure. Unlike previous methods, our algorithm systematically evaluates the interplay between the clique and independent set, allowing for an exact determination of the clique number with reduced computational complexity. By focusing on the distinct properties of split graphs, our approach outperforms existing methods, particularly in terms of computational efficiency.

This paper builds upon the groundwork laid by previous studies in both general graph theory and specialized algorithms for structured graph classes. It contributes a novel, efficient solution to the clique number problem in split graphs, demonstrating the benefits of structural exploitation and providing a new tool for graph analysts working with this important graph class.

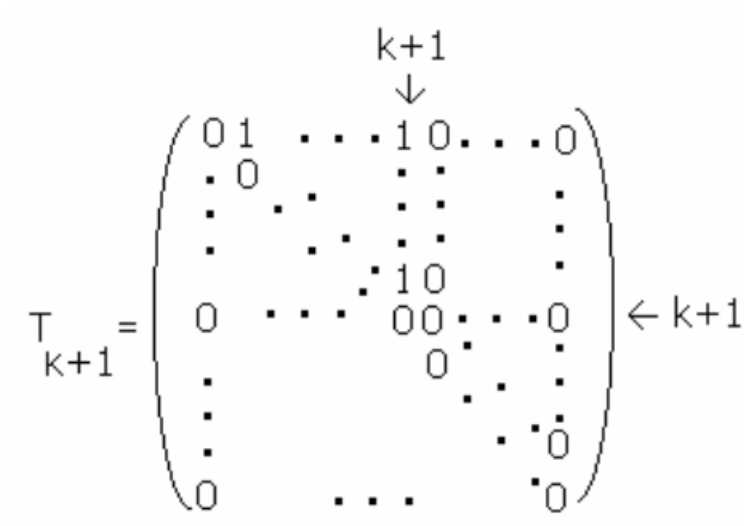
III. Method

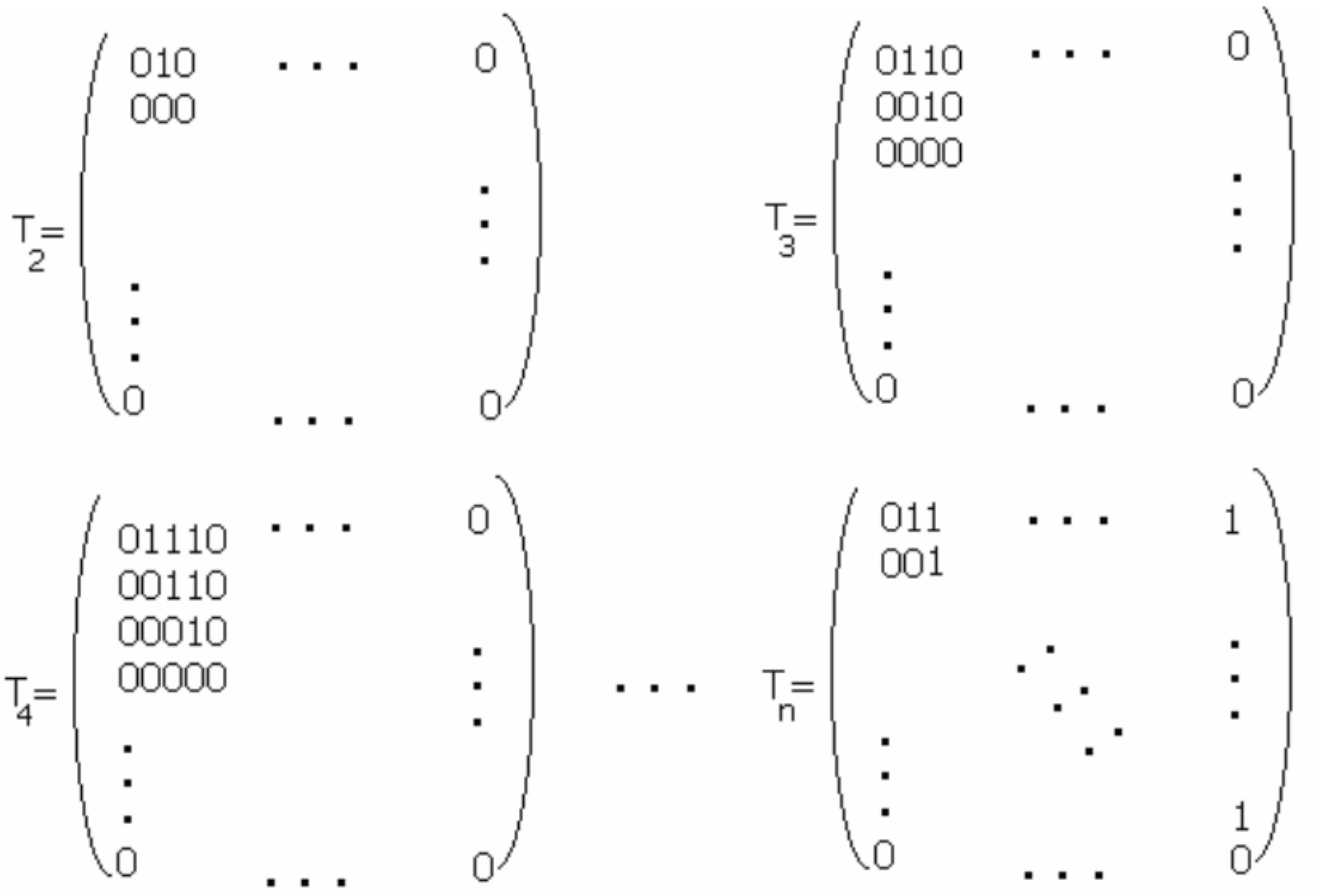
Definition

Let n be an integer such $n \geq 3$, for k such $1 \leq k \leq n-1$, we define the nxn matrices T_{k+1} by :

$(T_{k+1})_{i,j}=1$ if $1 \leq i \leq k$ and $2 \leq j \leq k+1$ and $i<j$

$(T_{k+1})_{i,j}=0$ otherwise.

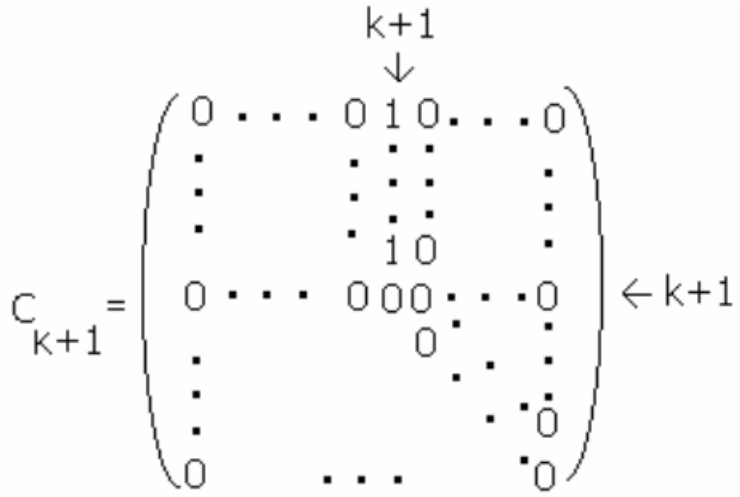




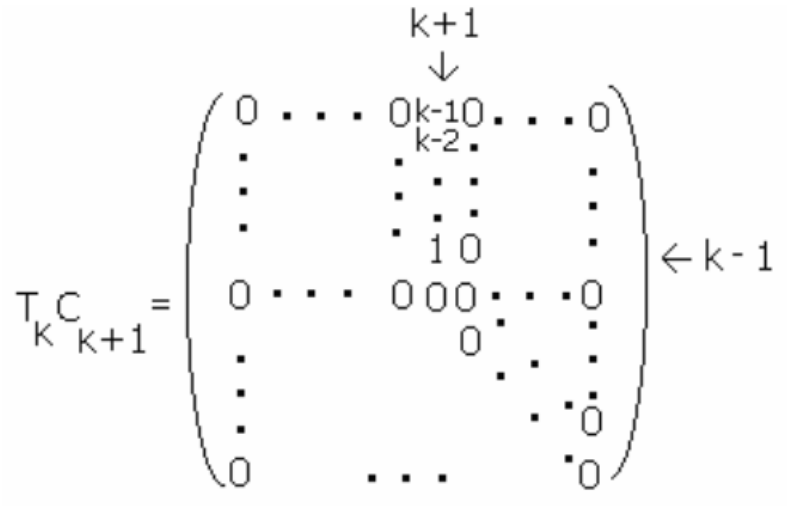
Proposition 1 :
For k such $1\leq k\leq n-1$, $(T_{k+1})^{k+1}=\theta$ and $(T_{k+1})^{h}\neq\theta$ if h is such $1\leq h\leq k$.
(where θ denote the nxn null matrix).

Proof :
For k such $1\leq k\leq n-1$, we define the matrix C_{k+1} by :
$(C_{k+1})_{i,j}=1$ if $j=k+1$ and $1\leq i\leq k$

$(C_{k+1})_{i,j}=0$ otherwise.



For k such $1\leq k\leq n-1$, we define the set of matrices M(k) by :
$M\in M(k)$ iff:
$(M)_{i,j}$ integer>0 if $j=k+1$ and $1\leq i\leq k$
$(M)_{i,j}=0$ otherwise.
For k such $2\leq k\leq n-1$, we have :
$T_{k+1}= T_k +C_{k+1}$
then
$(T_{k+1})^2=(T_k)^2+ T_kC_{k+1} +C_{k+1}T_k +(C_{k+1})^2$
since $(C_{k+1})^2=\theta$ and $C_{k+1}T_k=\theta$ and $T_kC_{k+1}\in M(k-1)$



then
$(T_{k+1})^2=(T_k)^2+ T_kC_{k+1}$ and $T_kC_{k+1}\in M(k-1)$
$(T_{k+1})^3=(T_k)^3+ T_k(T_kC_{k+1})$

$T_kC_{k+1}\in M(k-1)\Rightarrow T_k(T_kC_{k+1}) \in M(k-2)$
then:
$(T_{k+1})^3=(T_k)^3+ (T_k)^2C_{k+1}$ and $(T_k)^2C_{k+1}\in M(k-2)$
Suppose by induction on p that for p such $k>p\geq 1$:
$(T_{k+1})^p=(T_k)^p+ (T_k)^{p-1}C_{k+1}$ and $(T_k)^{p-1}C_{k+1}\in M(k-p+1)$
then
$(T_{k+1})^{p+1}=(T_k)^{p+1}+ T_k(T_k)^{p-1} C_{k+1}$ and $T_k(T_k)^{p-1}C_{k+1}\in M(k-p)$
Then for p such $k>p\geq 1$, we have :

$(T_{k+1})^{p+1}=(T_k)^{p+1}+(T_k)^pC_{k+1}$ and $(T_k)^pC_{k+1}\in M(k-p)$
for p=k-1 , we have :
$(T_{k+1})k=(T_k)k+(T_k)^{k-1}C_{k+1}$ and $(T_k)^{k-1}C_{k+1}\in M(1)$
since $(T_k)^{k-1}C_{k+1}\in M(1)\Rightarrow T_k(T_k)^{k-1}C_{k+1}=\theta$

then
$(T_{k+1})^{k+1}=(T_k)^{k+1}+(T_k)^kC_{k+1}$
then
$(T_{k+1})^{k+1}=(T_k)^{k+1}$ (1)
$(T_2)^2=\theta$
Suppose by induction on k that : $(T_k)^k=\theta$, then equation (1) implies :
$(T_{k+1})^{k+1}=\theta$
On the other hand, for p such$1\leq p\leq k$, we have:
$(T_{k+1})^p=(T_k)^p+(T_k)^{p-1}C_{k+1}$ and $(T_k)^{p-1}C_{k+1}\in M(k-p+1)$
since $1\leq p\leq k \Rightarrow k-p+1\geq 1 \Rightarrow (T_k)^{p-1}C_{k+1}\neq\theta$ and $(T_{k+1})^p\neq\theta$ for p such $1\leq p\leq k$.
Then for p such $1\leq p\leq k$, we have:
$(T_{k+1})^p\neq\theta$.

Now, for k such $1\leq k\leq n-2$, we define the matrices B_{k+1} by :
$(B_{k+1})_{i,j}\in\{0,1\}$ if $1\leq i\leq k$ and $k+2\leq j\leq n$
$(B_{k+1})_{i,j}=0$ otherwise.
For k such $1\leq k\leq n-1$, we define the set of matrices B(k) by :
$M\in B(k)$ iff:
$(M)_{i,j}\in\{0,1\}$ if $k+1<j\leq n$ and $1\leq i\leq k$
$(M)_{i,j}=0$ otherwise.

Then, we define the matrices S_{k+1} :
$S_{k+1}= T_{k+1}+ B_{k+1}$

Proposition 2 :
For k such $1\leq k\leq n-1$, $(S_{k+1})^{k+1}=\theta$
And
$(S_{k+1})^h\neq\theta$ if h is such$1\leq h\leq k$.

Proof :

Indeed :
$(S_{k+1})^2= (T_{k+1})^2+ (B_{k+1})^2+ B_{k+1}T_{k+1} +T_{k+1}B_{k+1}$
Since $(B_{k+1})^2=\theta$ and $B_{k+1}T_{k+1}=\theta$ and $T_{k+1}B_{k+1}\in B(k-1)$
then
$(S_{k+1})^2= (T_{k+1})^2+T_{k+1}B_{k+1}$ and $T_{k+1}B_{k+1}\in B(k-1)$
Suppose by induction on p such $1\leq p\leq k-1$, that :
$(S_{k+1})^p= (T_{k+1})^p +(T_{k+1})^{p-1}B_{k+1}$ and $(T_{k+1})^{p-1}B_{k+1}\in B(k-p+1)$
then
$(S_{k+1})^{p+1}=(T_{k+1})^p(T_{k+1}+ B_{k+1})= (T_{k+1})^{p+1} + (T_{k+1})^p B_{k+1}$ and $(T_{k+1})^p B_{k+1}\in B(k-p)$
Because $(T_{k+1})^{p-1}B_{k+1}\in B(k-p+1)\Rightarrow (T_{k+1})^p B_{k+1}\in B(k-p)$.
For p=k-1, we have :
$(S_{k+1})^k=(T_{k+1})^k+ (T_{k+1})^{k-1}B_{k+1}$ and $(T_{k+1})^{k-1}B_{k+1}\in B(1)$

And
$(T_{k+1})^{k-1}B_{k+1}\in B(1)\Rightarrow (T_{k+1})^k B_{k+1}=\theta$
Then
$(S_{k+1})^{k+1}=(T_{k+1})^{k+1}=\theta$
And

$(S_{k+1})^p= (T_{k+1})^p+(T_{k+1})^{p-1}B_{k+1}$ and $(T_{k+1})^{p-1}B_{k+1}\in B(k-p+1)$ for p such $1\leq p\leq k$,
Since, by proposition 1, $(T_{k+1})^p\neq\theta$ for p such $1\leq p\leq k$, then

$(S_{k+1})^p \neq \theta$ for p such $1 \leq p \leq k$,
we deduce that for k such $1 \leq k \leq n-1$, we have:
$(S_{k+1})^{k+1} = \theta$ and
$(S_{k+1})^h \neq \theta$ if h is such that $1 \leq h \leq k$.

Proposition 3:
Let G=(V,E) be a split graph on n vertices. Let U denote the upper triangular matrix extracted from the adjacency matrix of G. Then :
$\omega(G) = \min_{1 \leq h \leq n} \{h, \text{ such } (U)^h = \theta\}$

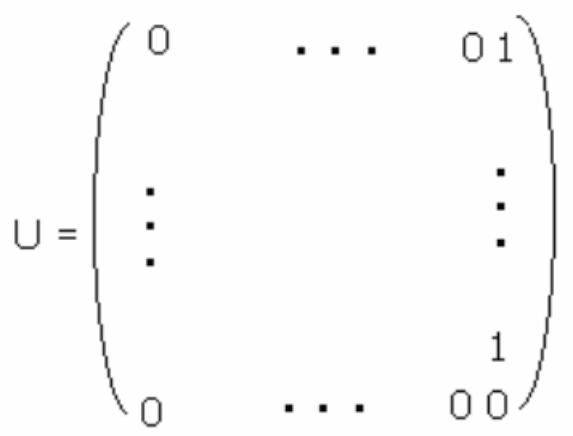
Proof :
If G contains a maximum clique of size k+1 , then U is congruent to a matrix S_{k+1} such :
$U = P^T S_{k+1} P$
were P is a permutation matrix and P^T is the transpose matrix of P.
then
$(U)^{k+1} = P^T (S_{k+1})^{k+1} P$

Then
$(U)^{k+1} = P^T \theta P = \theta$, by proposition 2.
and
$(U)^{k+1} = P^T (S_{k+1})^{h+1} P \neq \theta$ if h is such that $1 \leq h < k$, by proposition 2.
Then :
$k+1 = \min_{1 \leq h \leq n} \{h, \text{ such } (U)^h = \theta\}$
We deduce that:
$\omega(G) = \min_{1 \leq h \leq n} \{h, \text{ such } (U)^h = \theta\}$

Proposition 4:
Let G=(V,E) be a split graph on n vertices. Let U denote the upper triangular matrix extracted from the adjacency matrix of G. Then $\omega(G)$ can be computed by the following algorithm:
Input : U
1 k←1 and T←I
2 T←TU
if T=θ output(k) else k←k+1 goto step 2.

Proof :
By using proposition 3.
Note since matrix multiplication takes $O(n^3)$ time, the complexity of this algorithm is : $O(n^4)$.

Example 1 :
G is the complete graph K_n then $\omega(G)=n$.
Example 2 :
G is the graph $K_{1,n}$ then $\omega(G)=2$.



IV. Conclusion

In this paper, we presented a novel algorithm specifically designed to determine the clique number of split graphs, exploiting their unique structural properties. Split graphs, characterized by their partition into a clique and an independent set, provide a fertile ground for optimization, allowing the clique number problem to be solved more efficiently than in general graph classes. Our proposed algorithm systematically explores the interplay between the clique and independent set, ensuring an exact determination of the maximum clique size with reduced computational complexity.

Through theoretical analysis, we demonstrated that the algorithm operates in $O(n^4)$ time relative to the number n of vertices, significantly outperforming traditional methods that do not take advantage of the split graph structure.

The contributions of this work underscore the importance of developing specialized algorithms tailored to specific graph classes. By leveraging the inherent properties of split graphs, our approach not only provides an exact solution to the clique number problem but also sets the stage for further exploration of efficient algorithms for other structured graph classes.

Future research directions could include extending this approach to related graph classes, optimizing the algorithm for parallel and distributed computing environments, and exploring its applications in real-world problems where split graphs frequently arise, such as network analysis and data clustering. This work enhances our understanding of split graphs and provides a robust tool for graph theorists and practitioners working with large-scale graph data.

Reference

[1]. Földes, S., & Hammer, P. L. (1977). Split graphs. Congressus Numerantium, 19, 311-315.

[2]. Balasubramanian, S., & Rangan, C. P. (1994). Heuristics for maximum clique and minimum coloring on some perfect graphs. *Information Processing Letters*, *52*(2), 75-79.

[3]. Boros, E., Golumbic, M. C., & Levit, V. E. (2002). On the number of vertices belonging to all maximum cliques. *Discrete Mathematics*, *242*(1-3), 41-54.

[4]. Kojima, R., & Yamashita, T. (2018). Approximation algorithms for finding large cliques in split graphs. *Journal of Graph Algorithms and Applications*, *22*(3), 415-434.

[5]. Papadopoulos, C., & Tzimas, A. **(2017).** An efficient heuristic for the maximum clique problem on split graphs. *European Journal of Operational Research*, *257*(2), 545-553.