Application of Graph Theory and Variants of Greedy Graph Coloring Algorithms for Optimization of Distributed Peer-to-Peer Blockchain Networks

Miljenko Švarcmajer *, Denis Ivanović, Tomislav Rudec, Ivica Lukić *

Posted Date: 29 November 2024

doi: 10.20944/preprints202411.2383.v1

Keywords: greedy algorithms; graph coloring; DSATUR; distributed systems; peer-to-peer networks; connectivity optimization; private blockchain network; graph theory

Article

Application of Graph Theory and Variants of Greedy Graph Coloring Algorithms for Optimization of Distributed Peer-to-Peer Blockchain Networks

Miljenko Švarcmajer *, Denis Ivanović, Tomislav Rudec and Ivica Lukić

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

* Correspondence: miljenko.svarcmajer@ferit.hr

Abstract: This paper investigates the application of graph theory and variants of greedy graph coloring algorithms for the optimization of distributed peer-to-peer networks, with a special focus on private blockchain networks. The graph coloring problem, as an NP-hard problem, presents a challenge in determining the minimum number of colors needed to efficiently allocate resources within the network. The paper deals with the influence of different graph density, i.e. the number of links, on the efficiency of greedy algorithms such as DSATUR, Descending and Ascending. Experimental results show that increasing the number of links in the network contributes to a more uniform distribution of colors and increases the resistance of the network, whereby the DSATUR algorithm achieves the most uniform color saturation. The optimal configuration for a 100-node network has been identified at around 2000 to 2500 links, which achieves stability without excessive redundancy. These results are applied in the context of a private blockchain network that uses optimal connectivity to achieve high resilience and efficient resource allocation. The research findings suggest that adapting network configuration using greedy algorithms can contribute to the optimization of distributed systems, making them more stable and resilient to loads.

Keywords: greedy algorithms; graph coloring; DSATUR; distributed systems; peer-to-peer networks; connectivity optimization; private blockchain network; graph theory

1. Introduction

The question of the complexity of algorithms is one of the key ones in the theory of computing, and all problems that can be solved in polynomial time belong to class P [1]. On the other hand, the set of all problems for which we can check the correctness of the solution in polynomial time is denoted by NP. One of the most important open questions in the theory of computing is: is P = NP? Among the most famous NP problems is the problem of coloring graphs, which asks the question: with how many colors can we color the vertices of the graph so that no edge has ends of the same color? The smallest number of colors satisfying this condition is called the chromatic number of the graph [2].

Graph coloring is widely used in computer science, especially in optimization and resource scheduling. For example, when allocating registers in compilers, graph coloring enables efficient allocation of registers to variables in order to minimize the number of required registers and avoid collisions between variables [3]. In task scheduling, graph coloring helps to allocate tasks to a limited number of processors without conflicts, which is crucial for the optimization of parallel systems. Also, in network theory, graph coloring enables the allocation of frequencies to base stations in telecommunication networks to avoid interference between neighboring stations, thus optimizing spectrum utilization. It is also widely used in distributed computer networks [4] such as blockchain networks [5]. Furthermore, graph coloring is also used in exam scheduling, ensuring that students taking the same subjects do not overlap in terms [6].

One of the basic approaches to solving graph coloring problems is greedy algorithms, which work by assigning to each vertex the first available color that does not cause a conflict with

 doi:10.20944/preprints202411.2383.v1

neighboring vertices. Greedy algorithms are popular due to their simplicity and speed, which makes them suitable for application in situations where a fast response is required, such as task scheduling or register allocation [3,6]. However, they often use more colors than is optimal, making them suboptimal when trying to minimize the number of colors needed to color a graph [7].

The paper focuses on the application of graph theory and variants of greedy coloring algorithms to optimize connectivity and stability in distributed systems, with a special emphasis on private blockchain networks. The goal of the research is to identify the optimal connectivity configuration that allows for even resource distribution, reduced congestion, and increased network resilience to failures. The results have practical applications in the design of private blockchain networks, where the stability and efficiency of communication between nodes play a key role in maintaining a reliable and scalable network.

In the second chapter of this paper, the greedy algorithm will be described, and its basic algorithm will be given, and then the most frequently mentioned types will be processed. In the third chapter, related work will be described, that is, comparison of algorithms. In the fourth chapter, the results of testing subtypes of greedy algorithms on graphs of different density will be analyzed. Fifth chapter will describe how to apply graphs in network design following by sixth chapter which describes how to determine optimal number of links in computer networks using variants of greedy algorithms.

2. Greedy Algorithms

The graph coloring problem is NP-hard, which means that computing the optimal solution is very complex, especially for larger graphs [8]. One of the simplest and most widely used approaches to solving the graph coloring problem is the greedy algorithm. This algorithm processes vertices one by one, assigning them the smallest possible color that is not already used by neighboring vertices. The greedy algorithm for solving the graph vertex coloring problem works as follows:

1. The vertices of the graph are arranged in an arbitrary sequence.
2. The first peak in the series is colored with color 1.
3. Other vertices are colored with the color of the smallest number for which it is valid that this vertex is not connected to another vertex of the same color.
4. It returns to step 3 until all vertices are painted.

Although the greedy algorithm is simple, it often uses more colors than is optimal, which means that it does not always find the best solution. However, its speed and ease of use make it useful in many practical situations. In practice, the order in which the vertices are processed significantly affects the efficiency of the greedy algorithm. Some strategies include sorting the vertices by degree (the number of edges connecting them) in ascending or descending order or using a random order [3]. Sorting vertices in descending order is also called the Welsh-Powell algorithm [9]. In addition to this, the DSATUR algorithm is also popular, which colors vertices according to the degree of saturation, that is, it looks at how many neighboring vertices are already colored [10].

Descending (Welsh-Powell) Algorithm

Welsh-Powell is a variant of the greedy algorithm that uses descending order to color the vertices of a graph. The vertices are first sorted according to their degree, i.e. the number of neighbors they have, so that the vertices with the most neighbors are colored first. After the vertices are sorted, coloring takes place in the standard greedy way: each vertex is assigned the first available color that does not cause a conflict with its neighbors.

The advantage of this algorithm is that it often uses fewer colors than the basic greedy algorithm, because it colors the most connected vertices first, which reduces the number of colors needed for the rest of the vertices. Its main disadvantage is that it requires additional time to sort the vertices before coloring, but overall gives better results compared to random or unordered greedy algorithms [9].

 doi:10.20944/preprints202411.2383.v1

Ascending Algorithm

In the ascending greedy variant, the vertices are sorted according to the ascending order of degrees, that is, the number of neighbors each vertex has. The vertices with the smallest number of neighbors are colored first. After sorting, coloring takes place according to the standard greedy algorithm, where each vertex is assigned the first available color that does not cause a conflict with neighboring vertices.

The advantage of this variant is its simplicity and quick application. Vertices with a lower degree are processed first, which can facilitate coloring in specific situations. The disadvantage is using more colors than other variants because it leaves more complex vertices for later, when most colors are already occupied, which can lead to suboptimal coloring [11].

DSATUR Algorithm

The DSATUR (Degree of Saturation) algorithm also uses a greedy method but relies on the degree of saturation of the peak. The degree of saturation measures how many neighbors of a given vertex are already colored with different colors. In each step of the algorithm, the vertex with the highest degree of saturation is selected for coloring, i.e. the vertex with the most colored neighbors. If several vertices have the same degree of saturation, then the vertex with the highest degree (number of neighbors) is selected.

The advantage of this algorithm is that it often gives better solutions than other variants of greedy algorithms because it intelligently chooses vertices with the highest degree of saturation, which reduces conflicts and enables more efficient coloring. The disadvantage of this algorithm is that it requires more computation due to monitoring the degree of saturation, which makes it somewhat more complex and slower than classical greedy algorithms [10].

Random Algorithm

In this variant of the greedy algorithm, instead of sorting the vertices according to some strategy (such as the degree of the vertex in ascending or descending order), the vertices are colored in a random order. After the order of vertices is randomly selected, the algorithm proceeds with coloring using the classic greedy method: each vertex is assigned the first available color that does not cause a conflict with its neighbors. The advantage of this method is that it is very simple to implement and can quickly generate solutions. Due to the random selection of the order of vertices, the random greedy algorithm often gives suboptimal results and uses more colors than variants that use strategic ordering (eg Welsh-Powell) and this is the biggest flaw. However, due to its simplicity, it can be useful in situations where speed is more important than optimality [12].

3. Related Work

The graph coloring problem has wide applications in computer science, including resource optimization, task scheduling, and frequency allocation. Over the years, scientists have developed numerous methods and algorithms to improve the efficiency of the solution. The research of different approaches and variants of greedy algorithms is a key part of this work.

Some papers, such as [13] from 2013, focus on developing new heuristics for specific types of graphs, such as interval graphs. This paper introduces an optimal greedy heuristic that ensures that graphs are colored with the optimal number of colors using a properly chosen coloring order. The authors conclude that such optimization enables better performance in specific graph structures, thereby significantly reducing the complexity of the algorithm in real scenarios. Similarly, [3] explores the complexity of graph partitioning using the greedy approach. The authors conclude that greedy methods are particularly useful in large and complex graphs where speed optimization is key and suggest their application in graphs with specific partitioning requirements.

Other papers, such as [14] from 2024, explore variants of greedy algorithms such as b-greedy and z-greedy. The authors conclude that these variants provide better results in graphs with a high degree of connectivity, where classical greedy algorithms are not efficient. These variants help in

 doi:10.20944/preprints202411.2383.v1

more accurate coloring of complex graph structures, where the balance between the number of colors and the density of vertices plays a key role. Also, the paper [15] from 2016 compares several variants of greedy algorithms, including DSATUR and Welsh-Powell. It was concluded that the DSATUR algorithm is most suitable for graphs with high density, while Welsh-Powell achieves the best results in situations where the speed of the algorithm is key.

While papers like [7] from 2021, provide a comprehensive comparison of various methods, including First Fit, DSATUR, and Welsh-Powell, and analyze their performance in different graphs. The authors conclude that DSATUR is the most efficient for complex graphs with high density, while the Welsh-Powell algorithm is the fastest for solving simpler graphs. This paper emphasizes the importance of choosing an appropriate algorithm in accordance with the characteristics of graphs, thereby contributing to the understanding of the relationship between graph structure and algorithm efficiency. On the other hand, the paper [16] offers an overview of different algorithms on graphs from the DIMACS benchmark, and concludes that DSATUR consumes the least colors, while Welsh-Powell is the fastest. The authors recommend using these algorithms depending on priorities such as time optimization or number of colors.

In addition to these studies, the paper [17] from 1992, offers a deeper insight into the limitations of greedy algorithms, especially in the context of finding complex structures such as cliques in large graphs. The authors conclude that greedy algorithms, although fast, fail to effectively detect such structures, which indicates the need for more complex methods in specific cases.

In conclusion, these papers provide a comprehensive overview of different variants of greedy algorithms, highlighting their applicability in different contexts, from specialized types of graphs, such as interval graphs, to more complex structures with a large number of vertices. The papers also emphasize the importance of comparison between algorithms in terms of efficiency and applicability on different types of graphs.

4. Testing Variants of Greedy Algorithm and Analysis of Results

To test the impact of graph density on the efficiency of graph coloring with greedy methods, 10 cases were used. In all 10 cases, a graph with 100 vertices was used, while the number of edges, i.e. their density, increased. The smallest number of edges is 500. In each subsequent iteration, the number of edges increased by 500, and this method of incrementing reached 4500. The largest number of edges between 100 vertices is 4950, and this is singled out as a special case.

The following variants of the greedy algorithm were used: Descending (Welsh-Powell), Ascending, Random and DSATUR. For each edge density, each variant was tested a thousand times. The average of the colors was calculated, and the results are included in Table 1.

Table 1. Coloring graph with greedy algorithm variants.

Number of vertices	Descending	Ascending	Random	Dsatur	Fastest	Slowest	Best /Worst difference
500	6,41	8,07	7,31	6,42	DESC	ASC	20,56%
1000	9,80	11,84	10,83	9,97	DESC	ASC	17,19%
1500	13,02	15,35	14,19	13,32	DESC	ASC	15,18%
2000	16,41	19,05	17,72	16,75	DESC	ASC	13,81%
2500	19,93	22,94	21,46	20,43	DESC	ASC	13,10%
3000	24,00	27,33	25,75	24,58	DESC	ASC	12,19%
3500	28,95	32,68	30,86	29,59	DESC	ASC	11,41%
4000	35,22	39,47	37,31	35,84	DESC	ASC	10,78%
4500	44,91	50,41	47,77	45,76	DESC	ASC	10,91%
4950	100	100	100	100	-	-	0,00%

After processing the data from the table, some of the test conclusions are presented graphically. On the graph shown in Figure 1.

 doi:10.20944/preprints202411.2383.v1

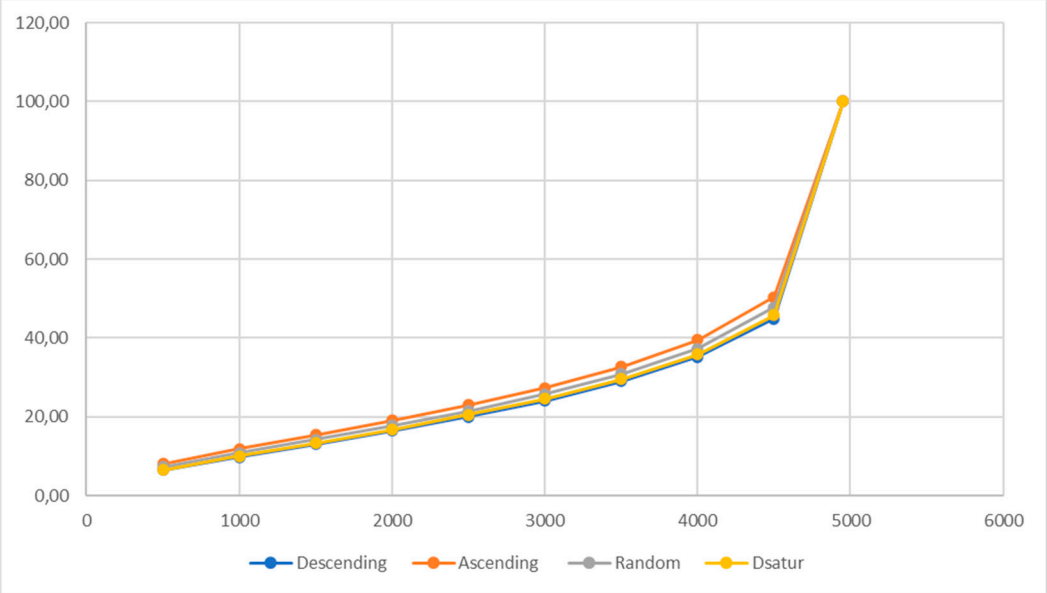

Figure 1. Impact of graph density on greedy algorithm variants.

Based on the presented graph, several key conclusions can be drawn about the influence of graph density on variants of greedy algorithms:

5. General trend and linear growth: All curves show an almost linear growth in the number of colors needed to color the graph as the number of edges increases, that is, as the graph becomes denser. This linear growth indicates that all algorithms have a proportional increase in the number of required colors with increasing graph density. As the number of adjacent vertices increases, more colors need to be used to avoid conflicts.
6. The Descending algorithm is the most efficient: The Descending algorithm consistently shows the best results compared to other variants. This means that sorting vertices in descending order, where the vertices with the largest number of neighbors are colored first, gives the best result because it minimizes the number of colors needed. This strategy is efficient because the most connected vertices receive colors before they are assigned to their neighbors, thus reducing the need for additional colors.
7. The Ascending Algorithm is the least efficient: The Ascending Algorithm almost always uses the largest number of colors. Sorting vertices in ascending order, where the vertices with the smallest number of neighbors are colored first, turns out to be an inefficient approach because it leaves more complex vertices for later, when most of the colors are already occupied.
8. Random and Dsatur algorithms: Random and Dsatur algorithms show average results, being placed between the Descending and Ascending variants. The dsatur algorithm is generally closer to Descending in efficiency because it uses the degree of saturation to select vertices, which allows it to better optimize the number of colors. The Random algorithm, although less consistent, also offers relatively good results.
9. Extreme density: It is interesting to note that all algorithms use the maximum number of colors (100% coloring) at a very high density of the graph, that is, when the graph has 4950 edges. This means that, in such a dense graph, each vertex must have its own unique color because all vertices are directly or indirectly connected.

Although the Ascending variant turned out to be the worst, its percentage reduction is still visible with regard to the best variant for a particular case. In the first variant, with the sparsest graph, the difference was 20.56%, while at the end of the test, with the densest graph (except for the special case with 4950 edges), that difference dropped to 10.91%. Figure 2 shows a drop in the percentage difference, but it can be seen that this drop is not linear.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2024 doi:10.20944/preprints202411.2383.v1

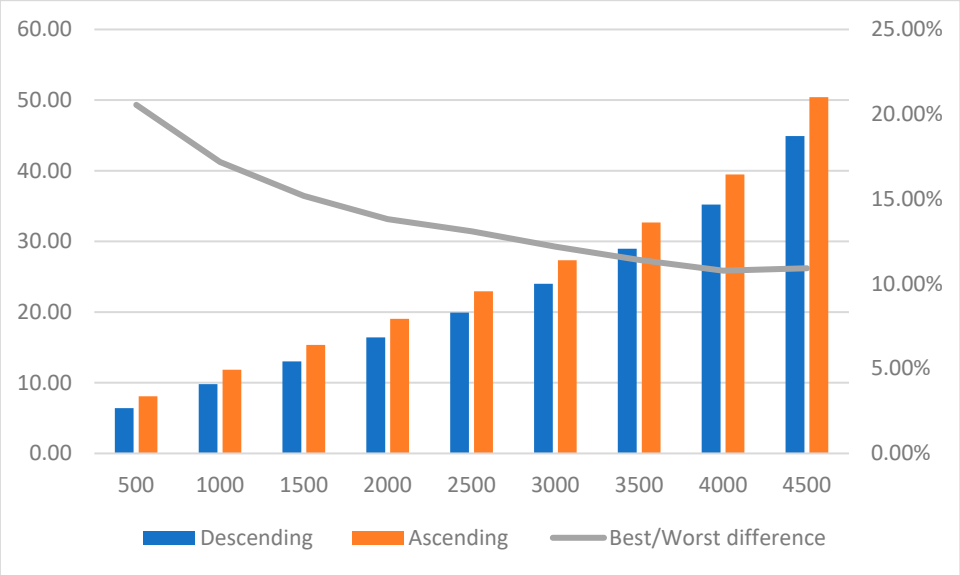

Figure 2. Percentage difference between best and worst case.

It can be concluded that the Descending algorithm is the most efficient in most cases, while the Ascending algorithm is the least efficient. Random and Dsatur algorithms offer a balance between speed and number of colors, depending on the density of the graph. The almost linear growth of the curves for all algorithms shows that the number of colors needed for graph coloring is proportional to the density of the graph, but also that the type of greedy algorithm can significantly affect the number of colors needed for efficient coloring.

5. Application of Graphs in Blockchain Network Design

The private distributed network of 100 nodes represents a blockchain system in which each node has the possibility of direct communication with other nodes, and the purpose of the network is to enable safe and efficient exchange of information. In the blockchain architecture, each node participates in the maintenance of a distributed database, ensuring high fault tolerance and preserving data integrity. This kind of network is designed as a peer-to-peer (P2P) system without a central server, which achieves decentralization and increases security [18].

In this chapter, it will be explained how graph theory and vertex coloring with variants of the greedy algorithm can help in modeling such a network. Nodes are represented as vertices, and connections between nodes as edges of the graph. The maximum number of edges between nodes is calculated according to the Equation (1):

$$Number\ of\ edges = \frac{n*(n-1)}{2} \quad (1)$$

The effect of the Number of Edges on the Performance of the Blockchain Network

In blockchain networks, there is no strictly defined minimum number of links (edges) that would automatically make the network decentralized and distributed. Decentralization in the blockchain architecture results from the independence of nodes and the distribution of data among nodes in the network. A network can be considered distributed and decentralized when nodes function independently of each other, data and tasks are evenly distributed, and nodes are sufficiently connected to allow information exchange and fault tolerance [20].

Problems with Too Few Links

Too few links in a network can significantly affect blockchain performance, creating communication bottlenecks, overloading certain nodes, and reducing fault tolerance. Namely, when

 doi:10.20944/preprints202411.2383.v1

nodes do not have enough links to other nodes, the risk of congestion increases because all information is transmitted through a limited number of channels, which reduces efficiency and increases network latency [21].

In a distributed network with too few links, the load becomes unevenly distributed, with some nodes overloaded by the need for multiple data transfers, while other nodes remain underutilized. This imbalance can compromise data security and integrity, as a lack of links makes it difficult to transfer data quickly and synchronize between nodes [20]. Figure 3 shows worst case scenario of the connected network. The minimum number of links was used in the network, which for this network of 100 nodes is 99. In the picture, the nodes with only one link are colored red. In such a network there is no redundancy and it is not resistant to failures.

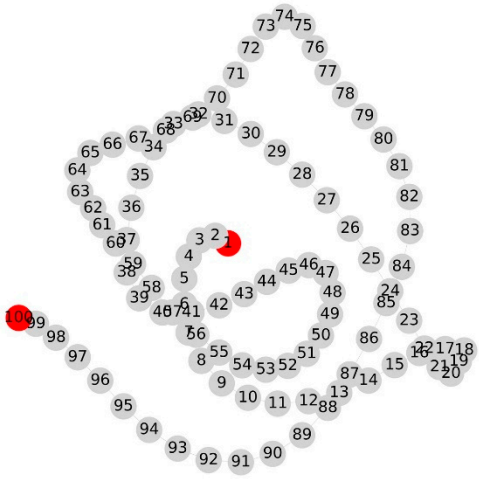

Figure 3. A network with 100 nodes and 99 links.

Advantages and Disadvantages of Too Many Links

The excessive number of links in the blockchain network has its advantages, but it also brings certain disadvantages. On the one hand, a larger number of links increases the network's resistance to failures because data can be redirected through alternative channels in the event of failure of certain nodes. More links enable better load balancing between nodes, which avoids bottlenecks and reduces data transfer latency [22].

However, networks with too many links increase the complexity of data routing and burden the system with unnecessary communications. Increased connectivity can also mean greater exposure to security risks, as more links provide more potential points of attack. In private blockchain networks, which require security and resilience, an excessive number of links can create additional security vulnerabilities and increase network maintenance costs due to the exponential growth of the number of links with the growth of nodes [20]. Figure 4 shows fully connected network with 100 nodes and 4950 links.

 doi:10.20944/preprints202411.2383.v1

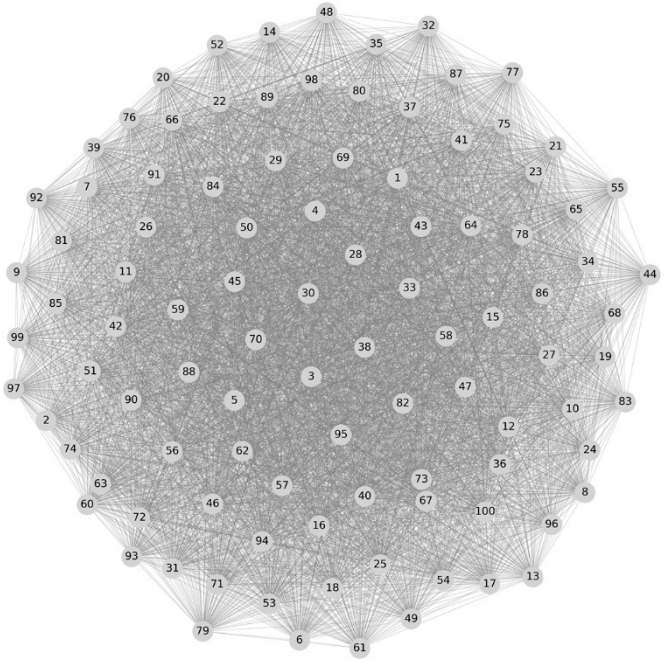

Figure 4. Fully connected network with 100 nodes and 4950 links.

Optimal Number of Links

In order to achieve the optimal number of links, it is necessary to balance network resilience with performance economy. Graph theory suggests that a network with a moderate number of links can maintain fault tolerance and security without unnecessary redundancy. Partial mesh or small-world topologies in practice enable a relatively small number of edges that ensure high resistance and efficiency of data transmission. In these topologies, the nodes are connected enough that data can circulate quickly through the network, but there are not an excessive number of edges to create congestion or increase complexity. The optimal number of links can vary. According to [23] and [24], for a network of 100 nodes, 4950 links are not needed. According to [25] an approximation is 30–50% of that number, which is about 1500–2500 links. This is a rough recommendation that often provides a good balance between resilience and avoiding excessive redundancy in large networks, but the exact optimal number of edges depends on the specific requirements of the network, such as security, throughput, and resilience. The next chapter will reduce this rough approximation to more concrete numbers.

6. Determining the Optimal Number of Links Using Variants of the Greedy Algorithm

Greedy graph coloring algorithms, such as Descending and DSATUR, can further optimize the distribution of links among nodes, allowing the allocation of links according to the current load and degree of connectivity of the nodes. In this way, the optimal configuration enables high resistance, efficient data transmission and reduces the possibility of creating bottlenecks.

Insights into Network Structure Properties from Graph Coloring Algorithms

In networks that use different coloring strategies, some variants of greedy algorithms have specific effects on the color distribution and structure of the network. In the DSATUR algorithm (Degree of Saturation), coloring starts with the nodes that have the highest saturation, which means that the nodes with the most different colors among their neighbors are colored first. This approach ensures that the most complex nodes are colored in a way that minimizes conflict in the network. If

 doi:10.20944/preprints202411.2383.v1

all the colors used in the DSATUR algorithm appear an equal number of times throughout the network, this means that each node has a similar number of neighbors with different colors, which suggests balanced connectivity and a consistent distribution of colors throughout the network. Such uniform color saturation helps stabilize the network and increases resilience to overloads, reducing the risk of bottlenecks.

The Descending Greedy algorithm colors the nodes with the most neighbors first, which means that the coloring focuses on densely connected parts of the network. This approach allows central nodes with a large number of links to receive colors that minimize conflicts, while less connected nodes are colored later. If all the colors used in the Descending algorithm occur equally, this means that the links in the network are evenly distributed. Uniformity in the repetition of colors in this case indicates that there are no nodes with a dominantly large number of one color, which allows the network to achieve a balanced degree of connectivity and avoid congestion in densely connected nodes. This pattern of color distribution contributes to the stability of the network, making it more resistant to overload and congestion.

The ascending greedy algorithm colors the nodes with the fewest neighbors first. Since these nodes have fewer links, they have a larger range of available colors, meaning they can be colored with fewer colors without conflicts. After coloring the nodes with fewer neighbors, the algorithm moves on to nodes with more neighbors, which then have a limited range of colors due to the already colored neighbors. The ascending approach results in a greater variety of colors in the edges of the network, while the central parts, with more links, remain balanced because they have enough options to avoid conflicts with the colors of their neighbors.

The application of the DSATUR algorithm ensures network stability because evenly distributed color saturation allows for balanced connectivity and reduces the risk of overload. The descending algorithm provides additional stability in networks with densely connected central parts because it allows for optimal color distribution among nodes with a high number of links. The ascending approach is useful for networks that require optimal color distribution in the edge parts, thus reducing the load on the central nodes. The combination of these algorithms can ensure network stability and resilience, adapting to different configurations and requirements of network systems, such as distributed and blockchain networks.

Coloring of a Network with 100 Nodes with Variants of the Greedy Algorithm for the Purpose of Optimization

For this paper, a computer-generated network of 100 nodes was used, and different amounts of edges were used. For this testing, a vertex represents a node in the blockchain network, while an edge acts as a link. Different link densities were tested.

1500 Links

Figure 5 shows a graph with 1500 links colored according to the DSATUR variation. Coloring by saturation shows diversity in the use of colors, but this concentration of colors is not uniform throughout the network. Nodes with higher saturation use specific colors at the beginning. Since such vertices are colored first, these colors are first in the list. The colors are relatively uniform, but there are parts with a concentration of specific colors, indicating uneven saturation of the network.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2024 doi:10.20944/preprints202411.2383.v1

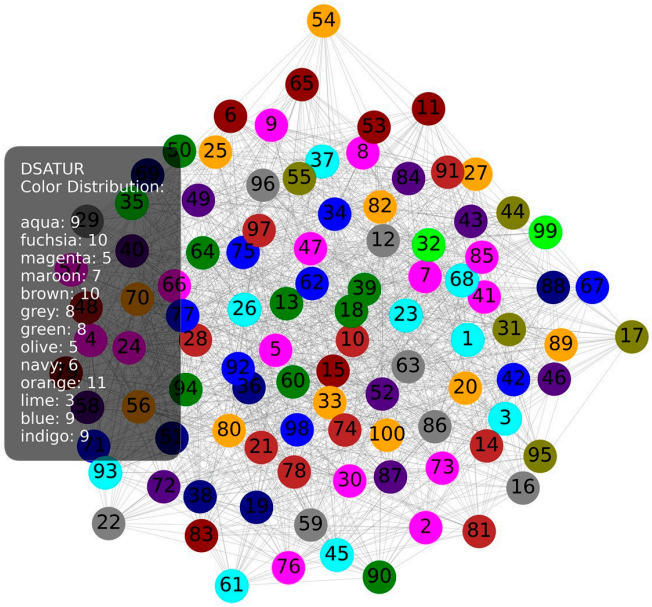

Figure 5. DSATUR 100 nodes 1500 links.

Figure 6, colored using the Descending variant, shows more colors in the very center of the network. Since this approach colors the nodes with the most neighbors first, it is evident that the nodes in the middle have more neighbors, that is, that the central nodes are more densely connected.

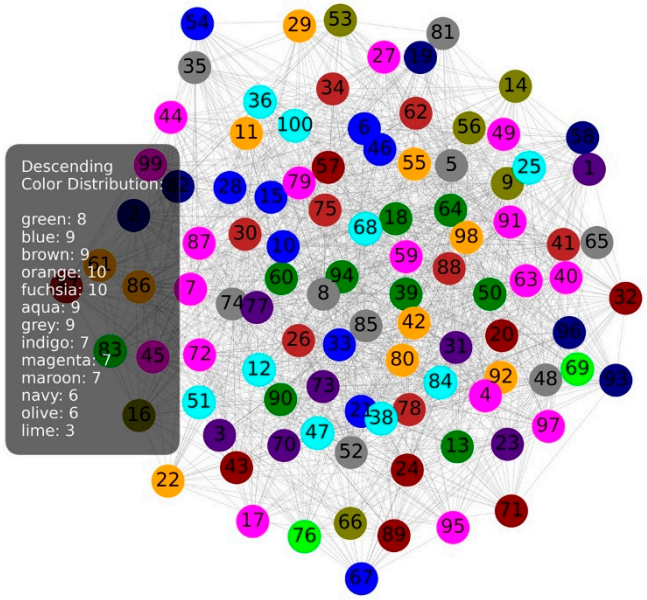

Figure 6. Descending 100 nodes 1500 links.

Figure 7 shows a larger number of different colors on the edge parts which can lead to a less uniform distribution in densely connected parts. In this example, the ascending variant of coloring was used.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2024 doi:10.20944/preprints202411.2383.v1

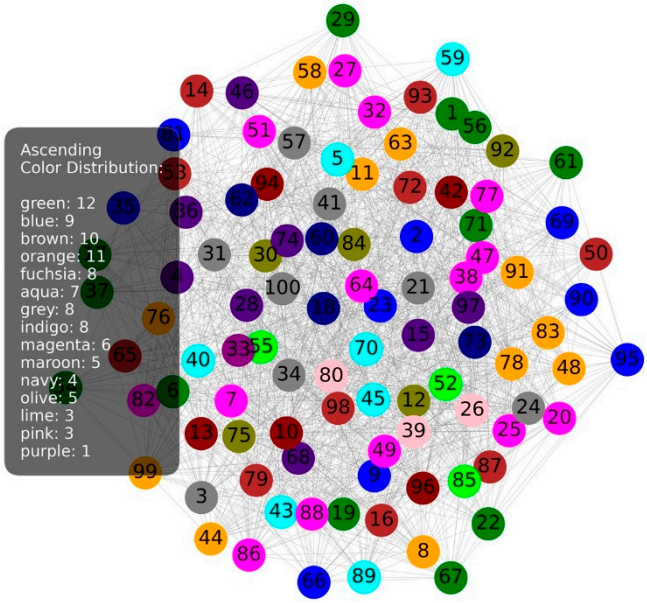

Figure 7. Ascending 100 nodes 1500 links.

These examples most clearly demonstrate the possibility of improving the network by pulling more links from peripheral nodes towards the center of the network.

2000 Links

As the number of links increases, the DSATUR approach in Figure 8 shows a more uniform distribution of colors throughout the network. This approach manages to keep the colors more evenly distributed in the central and peripheral parts of the network, indicating stability and resilience in the network.

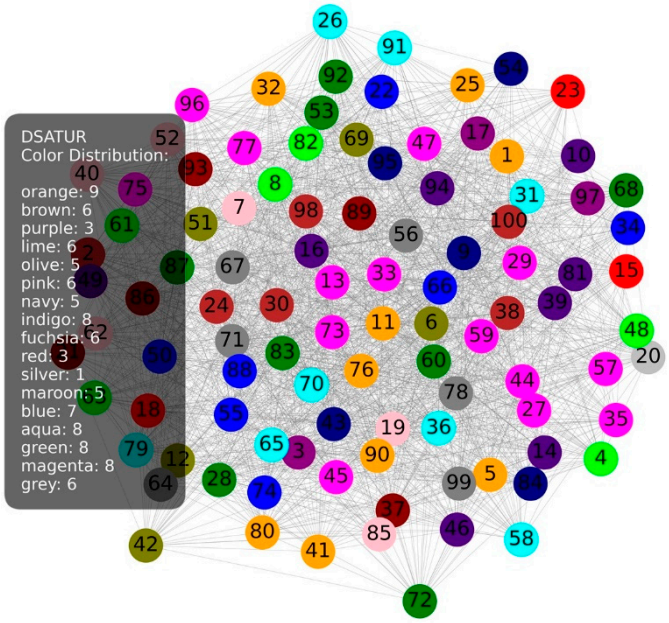

Figure 8. DSATUR 100 nodes 2000 links.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2024 doi:10.20944/preprints202411.2383.v1

The descending variant on Figure 9 shows greater color uniformity compared to the previous case. This approach now uses fewer colors in the central part of the network, which would mean that the network is more uniformly loaded.

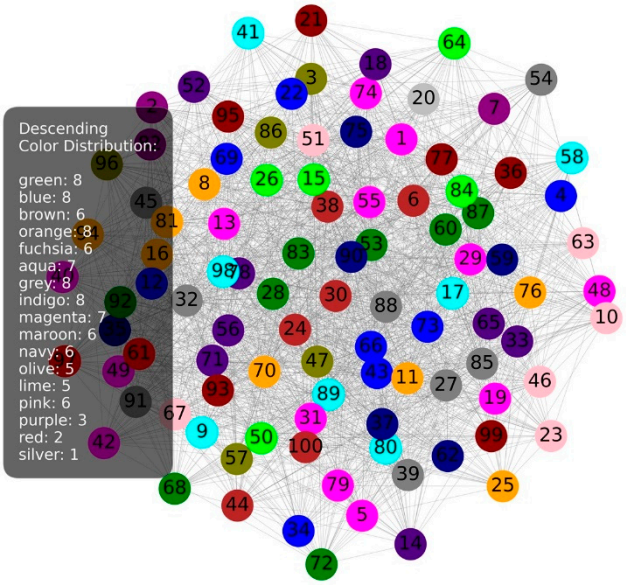

Figure 9. Descending 100 nodes 2000 links.

In Figure 10, the larger deviations in the use of colors at the edges of the network shown by the ascending variant are still visible. A greater variety of color usage on the peripheral parts indicates a weaker distribution towards those parts of the network.

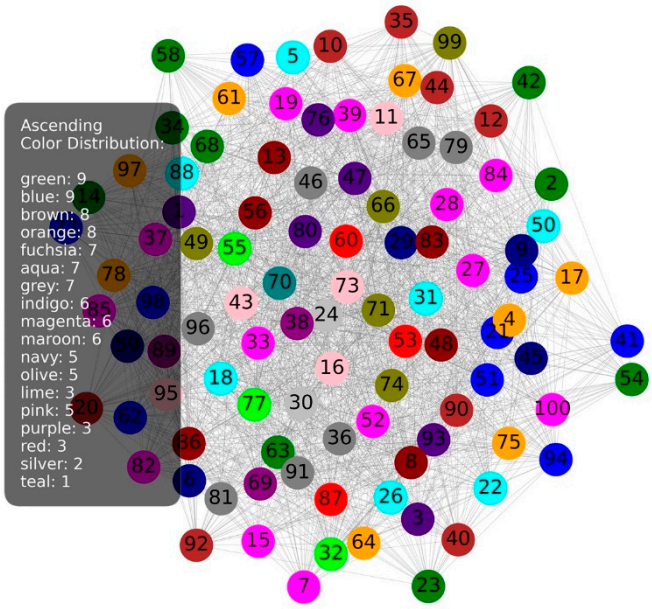

Figure 10. Ascending 100 nodes 2000 links.

When using 2000 links, slight improvements are visible in the form of less saturation of the central part of the network. The central part of the network is more evenly connected, which means

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2024 doi:10.20944/preprints202411.2383.v1

that very few nodes stand out with a higher number of links than others. The peripheral part of the network still shows weaker connectivity.

2500 Links

Increasing the number of links further improves the uniformity of coloring with the DSATUR algorithm. In Figure 11, the color is distributed even more evenly throughout the network, with balanced color usage in all parts. This case shows the highest stability.

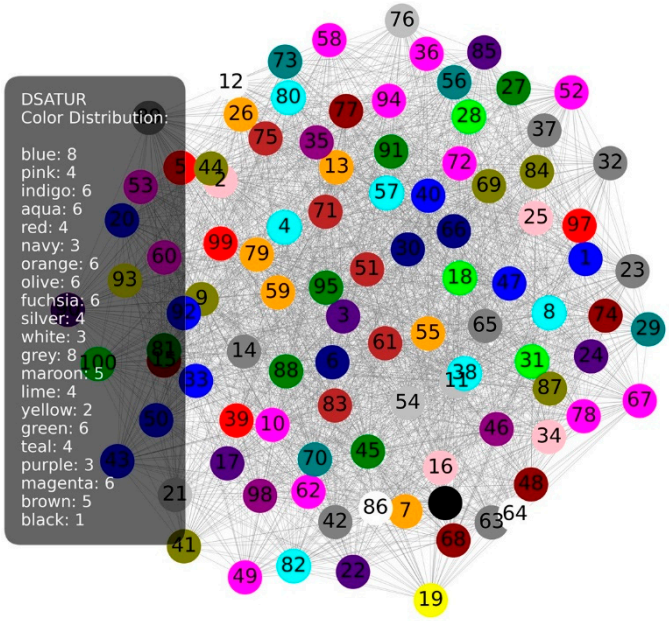

Figure 11. DSATUR 100 nodes 2500 links.

Increasing the number of links further improves the uniformity of coloring with the DSATUR algorithm. In Figure 12, the color is distributed even more evenly throughout the network, with balanced color usage in all parts. This case shows the highest stability.

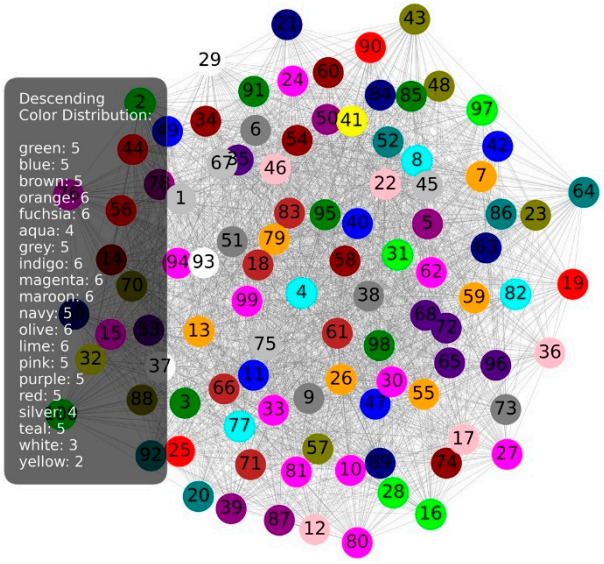

Figure 12. Descending 100 nodes 2500 links.

 doi:10.20944/preprints202411.2383.v1

The ascending variant in the case of Figure 13 also shows better uniformity in the color distribution compared to the smaller number of links. The colors are present in a larger number of edge parts, but now with reduced contrast between the central and edge parts of the network.

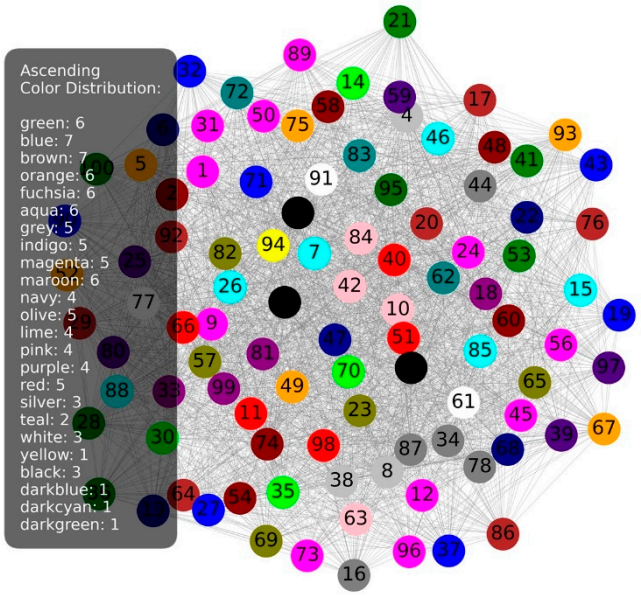

Figure 13. Ascending 100 nodes 2500 links.

Analysis of Results

Based on the analysis of each coloring approach for different numbers of links, several conclusions are drawn about the effects of using a larger number of links in the network and the optimal variants of the coloring algorithms. Using 2500 links results in the most balanced color distribution, especially when the DSATUR algorithm is applied. This approach achieves network stability and uniform color saturation among nodes, which indicates optimal connectivity without overloading individual nodes. In the DSATUR algorithm, all nodes have an equal number of neighbors with different colors, which reduces the risk of congestion and ensures stability.

A number of 2000 links can already provide enough connectivity for a 100-node network and be very close to optimal in many cases, especially when the goal is to balance resilience and efficiency. With 2000 links, the network should achieve high resilience and relatively good coloring uniformity, especially for algorithms like DSATUR and Descending.

From a coloring perspective, increasing to 2500 links does not provide a significant improvement over 2000 links. An additional 500 links may improve color uniformity somewhat, but this is usually a minimal improvement. Moreover, a larger number of links may increase redundancy and additional resource burden, without a significant positive effect on coloring stability.

Thus, the number of 2000 links can be considered an optimal compromise, providing high resistance and efficiency, while 2500 links can be an option for networks that require an additional safety net, but with minimal improvement in terms of coloring. This conclusion will be taken into account when creating the actual network.

7. Conclusions

This paper investigates the impact of graph density on the efficiency of greedy vertex coloring algorithms in solving the NP-hard graph coloring problem, with a special emphasis on applications in distributed peer-to-peer networks such as private blockchain systems. The results show that increasing the number of edges in the network can significantly improve the uniformity of the color

 doi:10.20944/preprints202411.2383.v1

distribution, with the DSATUR algorithm emerging as the most efficient for achieving stability and uniform color saturation.

The research found that increasing the number of edges from 1500 to 2000 led to significant improvements in stability and uniformity of coloring, making the network more resilient and less prone to overload. However, further increase to 2500 edges did not yield proportional improvements over the 2000-edge configuration, suggesting that 2000 edges are the optimal threshold for achieving a balance between connectivity and efficiency. This number of edges ensures network stability and sufficient resilience without excessive redundancy.

These results have practical value as they will be applied in a real private blockchain network that will use the optimal configuration of links and variants of greedy algorithms to adapt to network demands. Empirical testing of this configuration in a real system is expected to confirm the effectiveness of these conclusions, enabling high resilience, stability and efficient resource allocation in a private blockchain environment.

Author Contributions: Conceptualization, Tomislav Rudec and Miljenko Švarcmajer; methodology, Tomislav Rudec and Miljenko Švarcmajer; software Denis Ivanović and Miljenko Švarcmajer; validation, Tomislav Rudec and Ivica Lukić; formal analysis, Miljenko Švarcmajer; investigation, Miljenko Švarcmajer; resources, Denis Ivanović and Miljenko Švarcmajer; writing—original draft preparation, Miljenko Švarcmajer; writing—review and editing, Ivica Lukić, Tomislav Rudec, Denis Ivanović;; supervision, funding acquisition, Ivica Lukić. All authors have read and agreed to the published version of the manuscript."

Acknowledgments: This research was funded and is a part of research for activities to be carried out for project "Researching advanced algorithms and innovative business intelligence solutions in the cloud - NPOO.C3.2.R3-I1.04.0128."

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kapron, B.M. Logic, Automata, and Computational Complexity: The Works of Stephen A. Cook; Morgan & Claypool, 2023; ISBN 9798400707803.
2. Jeff Erickson Algorithms by Jeff Erickson; 2018;
3. Borowiecki, P. Computational Aspects of Greedy Partitioning of Graphs. J. Comb. Optim. 2018, 35, 641–665. https://doi.org/10.1007/s10878-017-0185-2.
4. Barenboim, L.; Elkin, M. Distributed Graph Coloring: Fundamentals and Recent Developments; Synthesis Lectures on Distributed Computing Theory; Springer International Publishing: Cham, 2013; ISBN 978-3-031-00881-8.
5. Jayabalasamy, G.; Pujol, C.; Latha Bhaskaran, K. Application of Graph Theory for Blockchain Technologies. Mathematics 2024, 12, 1133. https://doi.org/10.3390/math12081133.
6. Nurmalika, K.; Prihandini, R.; Jannah, D.; Makhfurdloh, I.; Agatha, A.; Wulandari, Y. Graph Coloring Application Using Welch Powell Algorithm On Radio Frequency In Australia. 2024.
7. Postigo, J.; Soto-Begazo, J.; Fiorela, V.R.; Picha, G.M.; Flores-Quispe, R.; Velazco-Paredes, Y. Comparative Analysis of the Main Graph Coloring Algorithms. In Proceedings of the 2021 IEEE Colombian Conference on Communications and Computing (COLCOM); May 2021; pp. 1–6.
8. NP-Hard Problem - an Overview | ScienceDirect Topics Available online: https://www.sciencedirect.com/topics/mathematics/np-hard-problem (accessed on 9 October 2024).
9. Olariu, S.; Randall, J. Welsh-Powell Opposition Graphs. Information Processing Letters 1989, 31, 43–46. https://doi.org/10.1016/0020-0190(89)90107-5.
10. San Segundo, P. A New DSATUR-Based Algorithm for Exact Vertex Coloring. Computers & Operations Research 2012, 39, 1724–1733. https://doi.org/10.1016/j.cor.2011.10.008.
11. Wang, Y. Review on Greedy Algorithm. Theoretical and Natural Science 2023, 14, 233–239. https://doi.org/10.54254/2753-8818/14/20241041.
12. Dobzinski, S.; Mor, A. On the Greedy Algorithm for Combinatorial Auctions with a Random Order 2015.
13. Olariu, S. An Optimal Greedy Heuristic to Color Interval Graphs. Information Processing Letters 1991, 37, 21–25. https://doi.org/10.1016/0020-0190(91)90245-D.

14. Costa Ferreira da Silva, J.; Havet, F. On B-Greedy Colourings and z-Colourings. Discrete Applied Mathematics 2024, 359, 250–268. https://doi.org/10.1016/j.dam.2024.08.001.
15. Aslan, M.; Baykan, N. A Performance Comparison of Graph Coloring Algorithms. International Journal of Intelligent Systems and Applications in Engineering 2016, 4, 1–1. https://doi.org/10.18201/ijisae.273053.
16. Wadsungnoen, B.; วาดส, เ.; Puphasuk, D.P. Permutation-Based Optimization Method for Solving Graph Coloring Problems.
17. Jerrum, M. Large Cliques Elude the Metropolis Process. Random Struct Algorithms 1992, 3, 347–359. https://doi.org/10.1002/rsa.3240030402.
18. (PDF) Blockchain for Committing Peer-to-Peer Transactions Using Distributed Ledger Technologies Available online: https://www.researchgate.net/publication/352455679_Blockchain_for_committing_peer-to-peer_transactions_using_distributed_ledger_technologies (accessed on 25 October 2024).
19. Wilson, R.J. Introduction to Graph Theory; Longman, 2010; ISBN 978-0-273-72889-4.
20. Wu, T.; Ren, H.; Li, P.; Leskovec, J. Graph Information Bottleneck. In Proceedings of the Advances in Neural Information Processing Systems; Curran Associates, Inc., 2020; Vol. 33, pp. 20437–20448.
21. Graph Theoretic Approaches for Analyzing Large-Scale Social Networks; Meghanathan, N., Ed.; 1st edition.; Information Science Reference: Hershey, Pennsylvania (701 E. Chocolate Avenue, Hershey, Pennsylvania, 17033, USA), 2017; ISBN 978-1-5225-2814-2.
22. Alon, U.; Yahav, E. On the Bottleneck of Graph Neural Networks and Its Practical Implications 2021.
23. Networks, Crowds, and Markets: A Book by David Easley and Jon Kleinberg Available online: https://www.cs.cornell.edu/home/kleinber/networks-book/ (accessed on 30 October 2024).
24. Graph Theory and Complex Networks: An Introduction: Van Steen, Maarten: 9789081540612: Amazon.Com: Books Available online: https://www.amazon.com/Graph-Theory-Complex-Networks-Introduction/dp/9081540610 (accessed on 30 October 2024).
25. Cohen, R.; Havlin, S. Complex Networks: Structure, Robustness and Function; 2010; ISBN 978-0-521-84156-6.