Acta Universitatis Sapientiae

Informatica

Volume 11, Number 2, 2019

Sapientia Hungarian University of Transylvania
Scientia Publishing House

**Acta Universitatis Sapientiae, Informatica
is covered by the following services:**

DOAJ (Directory of Open Access Journals)
EBSCO (relevant databases)
EBSCO Discovery Service
io-port.net
Japan Science and Technology Agency (JST)
Micosoft Academic
Ulrich's Periodicals Directory/ulrichsweb
Web of Science – Emerging Sources Citation Index
Zentralblatt für Mathematik

Contents

Acta Univ. Sapientiae, Informatica **11,** 2 (2019) 113–130

DOI: 10.2478/ausi-2019-0008

Evaluation metrics for anomaly detection algorithms in time-series

György KOVÁCS
Technical University of Cluj-Napoca, Romania
email: Gyorgy.Kovacs@cs.utcluj.ro

Gheorghe SEBESTYEN
Technical University of Cluj-Napoca, Romania
email: Gheorghe.Sebestyen@cs.utcluj.ro

Anca HANGAN
Technical University of Cluj-Napoca, Romania
email: Anca.Hangan@cs.utcluj.ro

Abstract. Time-series are ordered sequences of discrete-time data. Due to their temporal dimension, anomaly detection techniques used in time-series have to take into consideration time correlations and other time-related particularities. Generally, in order to evaluate the quality of an anomaly detection technique, the confusion matrix and its derived metrics such as precision and recall are used. These metrics, however, do not take this temporal dimension into consideration. In this paper, we propose three metrics that can be used to evaluate the quality of a classification, while accounting for the temporal dimension found in time-series data.

1 Introduction

Anomaly detection is the process of identifying erroneous data in big data sets, in order to improve the quality of further data processing. An anomaly detection method classifies data into normal and abnormal values. The selection

Computing Classification System 1998: G.2.2
Mathematics Subject Classification 2010: 68R15
Key words and phrases: anomaly detection, classification, evaluation metrics

of the best detection method greatly depends on the data set characteristics. Therefore, we need metrics to evaluate the performance of different methods, on a given data set.

Traditionally, in order to evaluate the quality of a classification, the confusion matrix, or one of its derived metrics is used. These metrics work well when the data set does not have a temporal dimension.

The anomaly detection task has certain particularities when it comes to time-series data. The temporal dimension that may be lacking in other types of data sets can be taken into account in order to improve the evaluation of these methods.

In this paper we propose some evaluation metrics which are more appropriate for time series. The basic idea of the new metrics take into consideration the temporal distance between the true and predicted anomaly points. This way, a small time shift between the true and the detected anomaly is considered a good result as opposed to the traditional metric that will consider it an erroneous detection.

Through a number of experiments, we demonstrate that our proposed metrics are closer to the intuition of a human expert.

The remainder of this paper is organized as follows: Section 2 discusses how time-series classification is used in the field and what metrics are used to evaluate the quality of classifications. Section 3 discusses the notation that we will be using in this paper going forward. The anomaly detection problem is defined for time-series data. We also define some prior requirements that we expect to be true for a time-series data metric that takes temporal distances into account. Section 4 presents the classification metrics we propose which are evaluated in Section 5. We check if the assumptions from Section 2 hold for our metrics. We also compare them with the traditional confusion matrix derived metrics such as accuracy, precision, and recall. We also present the result of applying our methods to real-world data. Section 6 concludes the paper.

2 Related work

Anomaly detection in time-series data is an important subset of generic anomaly detection. Much work has been done in developing anomaly detection methods. Some work has also been done in developing better metrics.

In many applications, it is more efficient to do feature extraction on the time-series data, and do classifications based on those features rather than on the actual time-series, as is the case for [7]. This is due to the fact the in many applications the volume of time series data is large and multi-dimensional. It is not easily analyzed and in many applications where speed is important, it is not practical to run algorithms directly on the raw data. Instead, the time series is split into segments, and for each segment features such as mean value, maximum and minimum amplitude and so on are calculated. Here classical clustering methods such as K-Nearest Neighbor can be used to classify each segment of time-series data.

The confusion matrix is generally used in the context of times series classification, which is the case in [1]. In [3] the authors use the confusion matrix explicitly as an input to train the classification model.

Better metrics for time-series have been proposed. In [2] the authors propose a metric that can differentiate between the generative processes of the time-series data. In [4] the authors propose a number of metrics such as Average Segmentation Count (ASC), Absolute Segmentation Distance (ASD) and Average Direction Tendency (ADT). These metrics were developed for evaluating a segmentation of a time-series, but they can be used just as well for evaluating the quality of anomaly detection. We will slightly modify the names of ASC and ASD by replacing segmentation with detection. In the experiments section, we will use these metrics Average Detection Count (ADC) and Absolute Detection Distance (ADD) and compare them with our metrics.

3 Problem statement

3.1 Notation

In order to express concisely the ideas presented in this paper, we will define the main concepts of anomaly detection and use the notation presented in this chapter for the following chapters as well.

This paper discusses concepts related to time-series data. By time-series data we mean an ordered set of real values that are indexed by natural numbers. We will not be discussing continuous values, since in practice we measure by sampling.

$$X = \{x_0, x_1, x_2, \dots, x_n\}, x_t \in \mathbb{R}$$

The main focus of this paper are classifications. The set of class labels, which will be referred to as a *classification*, is similar to X, the difference being that while X consists of real values, C consists of binary values $\{0, 1\}$. We will consider values labelled as 0 as being normal values, and values labelled as 1 being anomalous values.

$$C = \{c_0, c_1, c_2, \dots, c_n\}, c_t \in \{0, 1\}$$

The classification is generated by a classifier function $\mathcal{C}$. The classifier function takes an $x_t \in X$ value, with a range of size w around it, and generates a classification value c_t. Thus, this classifier can be used for a sliding window classification.

$$\mathcal{C} : X^{2w+1} \to C$$

$$c_t = \mathcal{C}(x_{t-w}, \dots, x_t, \dots, x_{t+w})$$

For a simplified example, consider the following time-series data:

$$X = \{8, 8, 8, 8, 42, 8, 8, 8, 8\}$$

We classify this data using the following classifier:

$$\mathcal{C}(x_t) = \begin{cases} 1 & \text{if } x_t > 10 \\ 0 & \text{otherwise} \end{cases}$$

In the example given $w = 0$. Thus the classifier only looks at one point for each classification. The result is the following classification:

$$C = \{0, 0, 0, 0, 1, 0, 0, 0, 0\}$$

We can represent the classification visually in Figure 1 as a line, where each point represents a classification. At the points where the classification value is 1, we draw a short vertical line through the horizontal line. We use this notation because our base assumption is that anomalous values are a disproportionately small subset of the whole set of values.

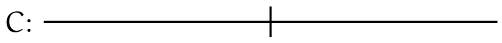


Figure 1: A classification C represented by a line with vertical lines where the value of C are 1.

Next, we define the classification evaluation problem. Supervised learning is a machine learning task with the purpose of reproducing a function by looking at example inputs and outputs. Given two or more potential candidate functions it is important for such an algorithm to be able to rank such functions.

In order to decide which one approximates the original function better, some metric is used.

This problem can be expressed as a comparison of classifications generated by different classifiers. We consider the target classifier $\mathcal{C}_0$. This is the function that we would like to reproduce. Given a number of different classifiers $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$ we would like to find which one approximates $\mathcal{C}_0$ the most.

In order to do this, we compare the classifications generated by them given the same training data X. We will use the graphical representation from Figure 2.

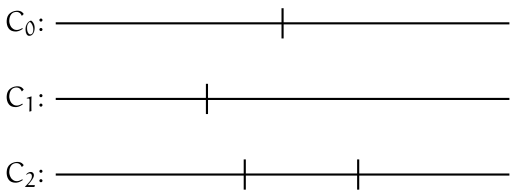


Figure 2: A comparison of three classifications, the first one being the target classification C_0 and the rest are regarded as the candidate classifications. One can see that C_1 identifies the anomaly prematurely while C_2 identifies two anomalies, one prematurely and one with a delay

We compare the classifications using a metric $m : \mathbb{C} \to \mathbb{R}$, $C_i \in \mathbb{C}$. The metric produces a score by comparing the given classification with the target classification. If a classification C_a is considered better than another one, say C_b, the metric would produce a better score for C_a. If a metric produces a better score for a classification, we consider that classification as better.

As an example, suppose we have the classifications from Figure 2. We will use a simple metric, that counts the number of anomalous points in each classification and computes the difference between that classification and the target classification.

We define the count function as the sum of all the elements of a classification. This effectively counts the number of anomalies because they are represented by the number 1, while the normal values are represented by the number 0.

$$\text{count}(C_i) = \sum_{j=1}^{n} c_j, c_j \in C_i$$

Next, we simply calculate the difference between the number of anomalies in the target classification and the candidate classification.

$$m_{\downarrow}(C_i) = |\text{count}(C_0) - \text{count}(C_i)|$$

Using this classification, we can see that the score for C_1 is $m_{\downarrow}(C_1) = 0$ and the score for C_2 is $m_{\downarrow}(C_2) = 1$. We can say that the first classification is better than the second one, since it has a lower value. This is represented by the subscript arrow that is pointing down. A metric where a higher value is better is denoted by a little arrow pointing up.

The examples presented in this chapter are simplistic and are only used to familiarize the reader with the notation that will be used for the remainder of this paper.

3.2 Prior requirements

We give examples with a target classification and a number of candidate classifications. We rank the classifications using our intuition. We would like a classification metric that can capture that intuition. These assumptions may or may not hold for certain applications.

Each of these situations will be tested both by the existing metrics used, and also our proposed metrics. In Section 5 we aggregate all the score data and show if the given metric does indeed respect these requirements.

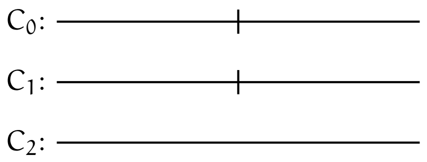
Detection The first requirement is to rank a classification that finds an anomaly higher than one that doesn't. The graphical representation can be seen in Figure 3a. Because C_1 correctly detects the anomaly, whereas C_2 does not.

False Detection The second requirement is that if there is indeed no anomaly, we would consider the classification that doesn't detect an anomaly as the better one. The graphical representation can be seen in Figure 3b. Because the target classification does not contain anomalies and C_2 falsely detects an anomaly, we can say that it is the worst from the two.

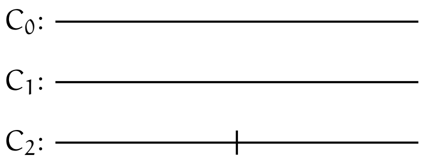
Less Wrong Whenever we have two classifications that correctly predict the anomaly, an ideal metric would choose the one with the fewer incorrect classifications, as presented in Figure 3c. Because both C_1 and C_2 correctly detect the anomaly, C_1 is considered better because it has less errors than C_2.

Near Detection Here we are starting to enter controversial territory. Given that we use time-series, we will hold the points near the point of interest in higher regard than those farther away. We consider that a classification that almost detects the anomaly correctly, is better than one that doesn't detect the anomaly at all. The graphical representation can be seen in Figure 3d. While none of the classifications manage to exactly detect the anomaly in the right place, we consider C_1 as better because at least it did detect something relatively close to the anomaly, while C_2 did not at all.

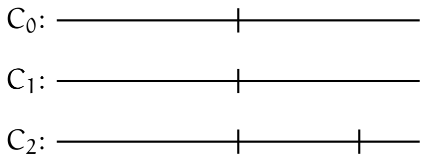
Closeness Going further, we consider that the closer the detection is to the actual anomaly in the target classification, the better that classification is. The graphical representation can be seen in Figure 3e. While both classification missed the anomaly, C_1 detected an anomaly closer to the target one than C_2.



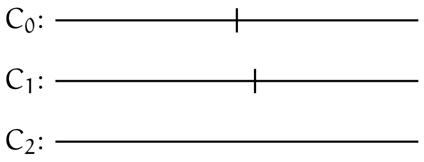
(a) Detection



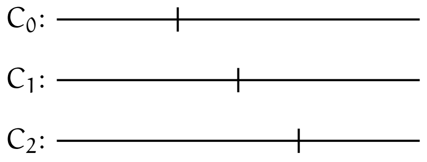
(b) False Negative



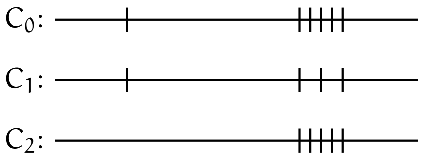
(c) Less Wrong



(d) Near Detection



(e) Closeness



(f) Locally Perfect vs Globally Good

Figure 3: Visualisation of the requirements. In each case the top classification (C_0) is the target classification and in these examples middle classification (C_1) is considered to be a better classification than the lower one (C_2).

Locally Perfect vs Globally Good Lastly, we introduce the principle of being Globally Good instead of Locally Perfect. This rule emphasizes the fact that we can have clusters of anomalies. Each cluster can have only one anomaly or multiple, but in close proximity to each other. This rule assumes that it is better to discover each cluster, rather than perfectly match every single anomaly from one single cluster. In the example from 3f, we can see that C_0 has two clusters, one with one single anomaly, and one with five close anomalies. We consider C_2 worse than C_1 even though it perfectly described the anomalies from the second cluster, because it failed to detect the first cluster.

4 Proposed metrics

4.1 Temporal distance method

This method consists of calculating the sum of all distances between anomalies from the two classifications. This method is similar to the ADD metric from [4]. The difference being that while in ADD we look only in the proximity of the detection, while our method looks at the closest detection, regardless of proximity. To this end we define a function that calculates the distance between each anomaly of the first classification and the corresponding closest anomaly from the second classification, $f_{closest} : \mathbb{C}^2 \to \mathbb{R}$.

Next we can define our method by using the function described above in the following manner:

$$TD_{\downarrow}(C_i) = TTC + CTT$$

where TTC stands for Target To Candidate and is given by $f_{closest}(C_0, C_i)$, and CTT stands for Candidate To Target and is given by $f_{closest}(C_i, C_0)$. C_0 stands for the target classification and C_i stands for the candidate classification.

Note that lower values produced by this method are better than higher ones. This is represented by a little downward pointing arrow.

We calculate both the sum of all the distances of the closest anomalies from the candidate classification to the target classification (CTT), and the sum of all distances of the closest anomalies from the target classification to the candidate one (TTC). By adding these two together, we have a metric that punishes false negative values and false positive values. TTC punishes false negatives and CTT punishes false positives.

A graphical visualization of the metric can be seen in Figure 4. We can see that C_0 has two anomalies, but C_1 only has one. Thus the closest anomaly to

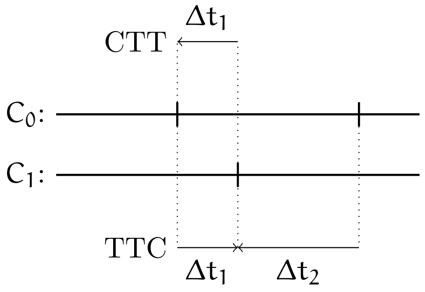


Figure 4: Visualization of the Temporal Distance metric. CTT $= \Delta t_1$ and TTC $= \Delta t_1 + \Delta t_2$

both anomalies from C_0 is the single anomaly in C_1. Because the two temporal distances are Δt_1 and Δt_2 respectively, TTC $= \Delta t_1 + \Delta t_2$. However, from the perspective of C_1, the closest anomaly to its only anomaly is the first from C_0. Only that one is taken into account, thus CTT $= \Delta t_1$. Finally the resulting value calculated by the metric is $\mathrm{TD}_{\downarrow}(C_1) = 2\Delta t_1 + \Delta t_2$. One can see that the best possible value for this metric is 0.

We also define a variation of this metric that we dubbed the Squared Temporal Distance. STD is defined similarly to TD, except when adding up the distances they are first squared. This is done in order to punish larger distances more than smaller ones. For example the value of STD for the given example is $2\Delta t_1^2 + \Delta t_2^2$.

4.2 Counting method

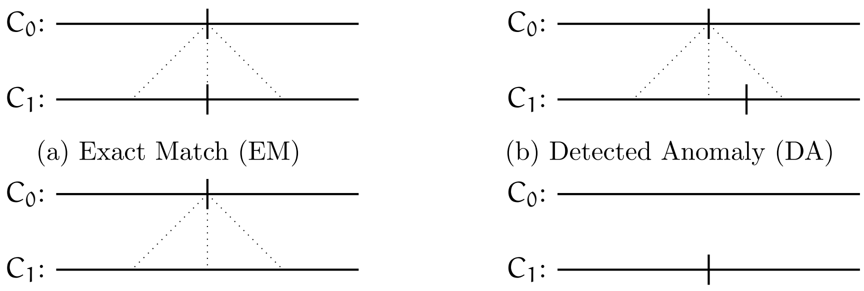
The next method is also similar to the ADC method defined in [4]. However our method is more analogous to a forgiving decision matrix, that counts close detections as true positives.

The counting method counts the occurrence of four situations. The situations of interest are pictured in Figure 5, and are as follows:

1. **Exact Match (EM)** Figure 5a. If the anomaly is from the candidate classification matches exactly the anomaly in the target classification, we call that an exact match.

2. **Detected Anomaly (DA)** Figure 5b. If the candidate anomaly does not match exactly the target anomaly, a range is considered. If the anomaly is withing that range, it is considered as a detection. However, that candidate anomaly will also be counted up as a false anomaly.

3. **Missed Anomaly (MA)** Figure 5c. If there is no anomaly present in the expected range of the target anomaly, we count it as a missed anomaly.
4. **False Anomaly (FA)** Figure 5d. Every normal target value that has an associated anomalous candidate value is counted as a false anomaly.



(a) Exact Match (EM) (b) Detected Anomaly (DA)

(c) Missed Anomaly (MA) (d) False Anomaly (FA)

Figure 5: Relevant situations that are each counted.

These four values can be used further to define derived metrics. We define two such metrics here:

Total Detected In Range We calculate the ratio of correctly detected anomalies to the number of total anomalies. With the maximum score of 1 and minimum of 0, this metric is similar to recall as derived from the confusion matrix. We can also call this metric “forgiving recall”.

$$\mathrm{TDIR}_{\uparrow} = \frac{\mathrm{EM} + \mathrm{DA}}{\mathrm{EM} + \mathrm{DA} + \mathrm{MA}}$$

Detection Accuracy In Range We calculate the ratio of correctly detected anomalies to the total number of detected anomalies. With the maximum score of 1 and minimum of 0, this metric is similar to precision as derived from the confusion matrix. We can also call this metric “forgiving precision”.

$$\mathrm{DAIR}_{\uparrow} = \frac{\mathrm{EM} + \mathrm{DA}}{\mathrm{EM} + \mathrm{DA} + \mathrm{FA}}$$

We could have implemented “forgiving” versions of the metrics defined for the confusion matrix, and we could have defined some new ones, like the ratio of exact matches to detected anomalies. However in the interest of conciseness, we will only consider the two defined above.

4.3 Weighted method

The weighted method can be seen as a combination of the previous two methods. For each anomaly of the target classification we calculate a weight that is given by a function that takes the distance of the candidate anomaly to the target anomaly and produces the weight. In Figure 6 we calculate a weight w for an anomaly. In the example a bell curve function is used. Any function based on distance can be used as long as it is monotonically decreasing.

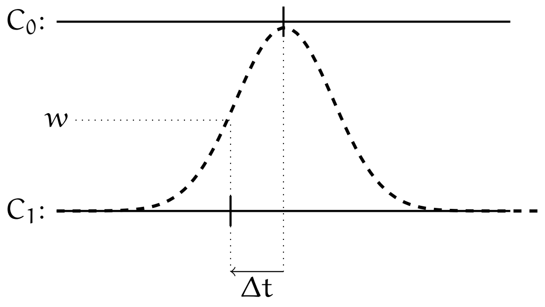


Figure 6: Weighted metric with a gaussian function, $w = f(\Delta t)$, where f describes a bell curve.

We denote with WS the sum up all the weighted values produced for each target anomaly. Note that we only take into consideration the closest candidate anomaly. We also count up all the false anomaly cases FA, similar to the previous section.

We define the **Weighted Detection Difference Metric** using the WS and FA. We just scale the FA by some factor and subtract it from the WS.

$$\mathrm{WDD}_{\uparrow} = \mathrm{WS} - w_f * \mathrm{FA}$$

where w_f is the weight of the false anomalies. Other functions were also considered such as a linear function:

$$f(\Delta t) = 1 - \frac{\Delta t}{t_{max}}$$

or a variant that punishes outliers equally:

$$f(\Delta t) = \begin{cases} 1 - \frac{\Delta t}{t_{max}} & \text{if } \Delta t < t_{max} \\ -1 & \text{otherwise} \end{cases}$$

5 Analysis and experiments

5.1 Requirements evaluation

We took each of the six rules defined in Section 3.2, and we turned it into synthetic data. We did this by considering 101 points for each Classification. For each point we assigned a one where the figure had a vertical line, and assigned a zero for the rest of the points. In fact the figures that appear in Section 3.2 where generated from these synthetic datasets. We used the synthetic datasets and calculated the scores produced for all the methods described in the previous chapter alongside classical methods. The results are aggregated in Table 1.

Table 1 is the cross product of the metrics we used and the rules we defined in Section 3.2. We differentiate between four possible outcomes. Each outcome is represented by a special character:

- ✓ We represent cases where the metric strictly respects the rule set out by us via a checkmark. Effectively this means that the metric produced a better result for the first candidate classification than for the second one. The actual value produced by the classification can be larger or smaller, depending on the metric used.
- = Equality does not respect the rule we set out in Chapter 2. However we decided to show it explicitly because it gives better insight into the workings of the metric. While we do not consider equality cases to be an instance of a successful quality evaluation, we consider them as cases where the metric can not tell the difference between two candidate classifications.
- × In cases where the metric gives a strictly better score to a worse candidate classification, we consider that as a broken rule. A metric that would fit the rules we laid out would never break any rules.
- - There are cases where a metric can not be calculated. Otherwise stated, there are classifications that yield scores that can not be expressed by

real numbers. These cases arise because the metric can use the number of anomalies detected in order to divide some other number. If there are no anomalies, we can not perform that operation. We call this situation undecidable and use a dash to denote that situation.

In the second example we produced the ranking in a similar fashion to the previous example. The actual change point happens in the third group of anomalies from C_0. We consider that only the change point is an anomaly. The rest of the anomalies are outliers. Outliers can be found both before and after the change point.

The metrics that best matched the imposed ranking this time were Recall, ADD, STD and DAIR. In this particular example the classical methods had similar distances to the proposed metrics. We believe that this is because of the fact that in this particular example, all of the anomaly groups were made up of sequential anomalies.

Consider a classifier that is always a few time-samples behind with the classification. If all anomalies are point anomalies, all candidate anomalies would miss the target anomalies and a classical metric would produce a bad score. Now if the anomalies were not point anomalies, but were a continuous intervals, even though the candidate intervals of anomalies were shifted, most anomalies would still overlap, thus producing a better score.

$\mathfrak{m}$	Det	FDet	LWrong	NDet	Close	LP vs GG
Accuracy$_\uparrow$	✓	✓	✓	×	=	×
Precision$_\uparrow$	✓	-	=	=	=	×
Recall$_\uparrow$	-	-	✓	-	=	=
ADC$_\downarrow$	✓	=	=	✓	✓	✓
ADD$_\downarrow$	-	-	=	-	✓	=
TD$_\downarrow$	-	-	✓	-	✓	✓
STD$_\downarrow$	-	-	✓	-	✓	✓
TDIR$_\uparrow$	✓	-	=	✓	✓	✓
DAIR$_\uparrow$	-	-	✓	-	✓	=
WDD$_\uparrow$	✓	✓	✓	✓	✓	✓

Table 1: For each rule defined in Section 2 we verify if the metric respects the relation.

The table shows that while some of the proposed metrics may sometimes be unable to distinguish between two classifications or evaluate an answer, they never give erroneous results. The same can not be said of Accuracy or Precision.

From our experiments, while WDD obtained the best results, the efficacy of the metric is very much dependent on the parameters used. By modifying the parameters we can get counter intuitive results.

5.2 Real data example

In order to further test the quality of our metric we will apply them to some example time-series traffic data. Two datasets will be considered. The first only has outlier points while the other also contains a change point.

Both datasets and classifiers are used in [8]. The classifiers are described in chapter 4 of that article. The classifications $\{C_1, \ldots, C_5\}$ are generated by the classifiers {Bounded Derivative (d = 0), Bounded Derivative (d = 1), Median Method, Linear Approximation, First Order AR}. The classifiers will not be discussed in this paper.

The first dataset is from [5], and the second one from [6]. The classification diagrams of the aforementioned datasets can be seen in Figure 7 and Figure 8 respectively. We evaluate each classification with each of the metrics used in Table 1. We also add another metric that ranks each classification according to the subjective opinions of the authors. We would like that all metrics generate a similar order to the one imposed by us.

The imposed ranking is done by assigning a number starting from 1 to each classification, where 1 is considered the best classification, and 5 is considered the worst. Next we compare that ranking with the ranking generated by the metrics. We consider the classification with the best result as the one with ranking of 1, the second best with 2 and so on. This step can be checked manually in the table. Finally we calculate the distance between the imposed ranking and the ranking generated by the metric. The lower the distance, the better the metric performed. The distance is calculated by summing the differences between the two rankings.

$$\text{Distance} = \sum_{i=1}^{5} |r_i - \hat{r}_i|$$

where r_i is the ranking of the classification C_i generated by a metric m and $\hat{r}_i$ is the imposed ranking of the given classification C_i.

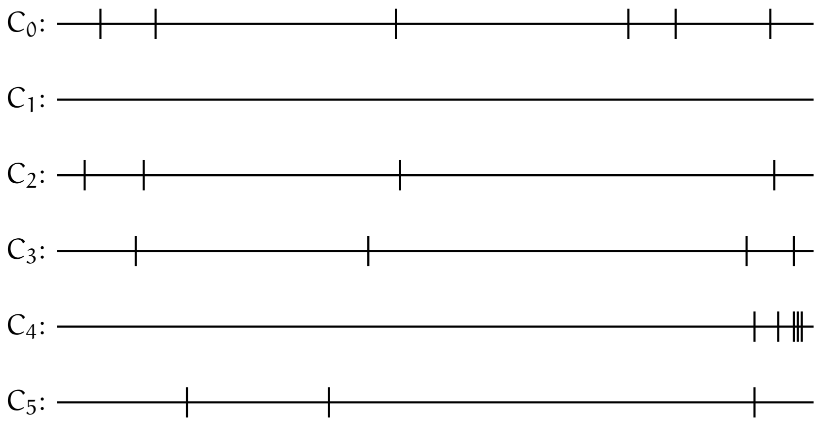


Figure 7: CO_2 emissions

$\mathfrak{m}$	$\mathfrak{m}(C_1)$	$\mathfrak{m}(C_2)$	$\mathfrak{m}(C_3)$	$\mathfrak{m}(C_4)$	$\mathfrak{m}(C_5)$	Distance
Accuracy$_\uparrow$	0.9688	0.9479	0.9479	0.9427	0.9531	9
Precision$_\uparrow$	0.0000	0.0000	0.0000	0.0000	0.0000	10
Recall$_\uparrow$	-	0.0000	0.0000	0.0000	0.0000	6
ADC$_\uparrow$	0	4	4	5	2	5
ADD$_\downarrow$	-	9	24	27	12	2
TD$_\downarrow$	-	8	99	490	132	0
STD$_\downarrow$	-	2048	1561	60538	2646	2
TDIR$_\uparrow$	0.0000	0.6667	0.6667	0.1667	0.3333	1
DAIR$_\uparrow$	-	0.5000	0.5000	0.1667	0.4000	1
WDD$_\uparrow$	0.0000	-3.5000	-3.9000	-43.3000	-6.1000	8
Ranking$\downarrow$	5	1	2	4	3	0

Table 2: CO_2 emissions. Note that all missing values are considered to be the worst scores.

For the first example we considered C_0. None of the classifiers managed to pin down the anomalies exactly. While all of them were off by some margin, some of them are clearly worse than others. For example, C_1 didn't detect any anomalies, while C_4 detected a cluster of them towards the end, where only one anomaly exists. Looking at the table of results, we can see that precision and accuracy can not tell the difference between these classifications. However, we would argue that C_2 is the clear winner. While the detection of the anomalies are off by one or two time samples, they are still in the neighborhood of the true anomalies. Only the fourth and fifth anomalies are missed by it. 2.

The metrics that best matched our ranking were TD, TDIR, DAIR. STD and ADD matched but they are also good classifications. This example shows the potential instability of the WDD method, that produced a ranking that is as bad as the ones produced by the confusion matrix.

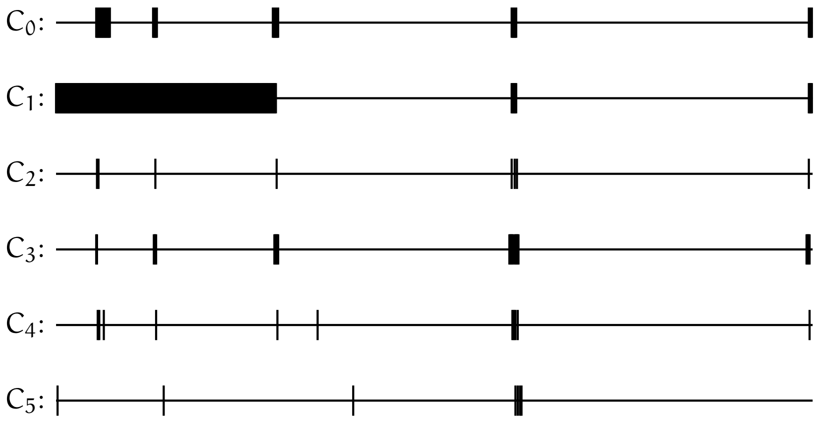


Figure 8: Concurrent users – Change point

6 Conclusion

In this paper we tackled with the problem of qualitative metrics applied to anomaly detection in time-series data. We concluded that classical metrics such as Accuracy, Precision and Recall do not take into consideration the time dimension of time-series data, in which near matches might be just as good as exact matches, or at least they are better than complete misses.

We defined the problem in more rigorous terms, and provided some requirements that we believe a good metric should meet. Next we defined some new metrics. We checked whether or not our proposed metrics respect the requirements set out by us previously. We also compared the performance of our

m	$m(C_1)$	$m(C_2)$	$m(C_3)$	$m(C_4)$	$m(C_5)$	Distance
Accuracy$_\uparrow$	0.7354	0.9609	0.9659	0.9619	0.9498	4
Precision$_\uparrow$	0.9333	0.1555	0.4444	0.2	0.0222	8
Recall$_\uparrow$	0.1386	0.875	0.6896	0.8181	0.1428	2
ADC$_\uparrow$	92	8	29	10	5	8
ADD$_\downarrow$	275	1	18	2	23	2
TD$_\downarrow$	8101	166	183	147	4123	4
STD$_\downarrow$	364933	1360	1849	3079	892305	2
TDIR$_\uparrow$	1	0.8888	0.8444	1	0.2	11
DAIR$_\uparrow$	0.1470	0.9756	0.8085	0.9574	0.6	2
WDD$_\uparrow$	-112.1999	27.8999	23.0999	34.4999	-353.6	6
Ranking$\downarrow$	5	1	2	3	4	0

Table 3: Concurrent users – Change point

proposed metrics with the performance of the classical metrics. We concluded that our metrics never gave an incorrect answer. The same could not be said of the classical methods.

We also applied our proposed metrics to two real datasets of web traffic. We compared the performance of all metrics discussed, and concluded that our proposed metrics performed better or the same as the classical metrics.

References

[1] E. Baidoo, J. Lewis Priestley,An Analysis of Accuracy using Logistic Regression and Time Series, *Grey Literature from PhD Candidates.* **2** (2016), `https://digitalcommons.kennesaw.edu/dataphdgreylit/2/`. ⇒115

[2] J. Caiado, N. Crato, D. Peña, A periodogram-based metric for time series classification, *Computational Statistics Data Analysis* **50** (2006) 2668–2684. ⇒ 115

[3] B. Esmael, A. Arnaout, R. K. Fruhwirth, G. Thonhauser, Improving time series classification using Hidden Markov Models, *Proceedings of the 12th International Conference on Hybrid Intelligent Systems (HIS)*, 2012, pp. 502–507. ⇒115

[4] A. Gensler, B. Sick, Novel Criteria to Measure Performance of Time Series Segmentation Techniques, *Proceedings of the LWA 2014 Workshops: KDML, IR, FGWM*, Aachen, Germany, 2014. ⇒115, 120, 121

[5] R.J. Hyndman, Time Series Data Library, Accessed: 2018-11-12, `https://datamarket.com/data/list/?q=provider:tsdl`. ⇒126

[6] N. Laptev, S. Amizadeh, I. Flint, Generic and Scalable Framework for Automated Time-series Anomaly Detection, *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 1939–1947. ⇒126

[7] S.I. Lee, C.P. Adans-Dester, M. Grimaldi, A.V. Dowling, P.C. Horak, R.M. Black-Schaffer, P. Bonato, J.T. Gwin, Enabling stroke rehabilitation in home and community settings: a wearable sensor-based approach for upper-limb motor training, *IEEE journal of translational engineering in health and medicine* **6** (2018) 1–11. ⇒115

[8] Gh. Sebestyen, A. Hangan, Gy. Kovacs, Z. Czako, Platform for Anomaly Detection in Time-Series, *XXXIV. Kandó Conference*, Budapest, Hungary, 2018. ⇒ 126

Received: September 17, 2019 • Revised: November 26, 2019

Acta Univ. Sapientiae, Informatica **11,** 2 (2019) 131–141

DOI: 10.2478/ausi-2019-0009

Metric space method for constructing splitting partitions of graphs

Sándor SZABÓ
University of Pécs, Pécs, Hungary
email: `sszabo7@hotmail.com`

Abstract. In an earlier work [6] the concept of splitting partition of a graph was introduced in connection with the maximum clique problem. A splitting partition of a graph can be used to replace the graph by two smaller graphs in the course of a clique search algorithm. In other words splitting partitions can serve as a branching rule in an algorithm to compute the clique number of a given graph. In the paper we revisit this branching idea. We will describe a technique to construct not necessary optimal splitting partitions. The given graph can be viewed as a metric space and the geometry of this space plays a guiding role. In order to assess the performance of the procedure we carried out numerical experiments.

1 Introduction

Throughout this note the word graph is used for in the restricted meaning of finite simple graph, that is, each graph will have finitely many vertices and finitely many edges. Further, neither loops nor double edges may occur. Let $G = (V, E)$ be a finite simple graph, where V is the node set and E is the edge set of G. The set of edges E consists of unordered pairs of elements of V. The simplicity of the graph G means that it has neither double edges nor loops. The finiteness of the graph G means that the sets V and E have finitely many elements.

Computing Classification System 1998: G.2.2
Mathematics Subject Classification 2010: 05C15,05B45, 52C22
Key words and phrases: clique number, maximum clique, branching rules.

A subgraph Δ of G is called a clique in G if two distinct nodes of Δ are always adjacent in G. If the clique Δ has k nodes we will say that Δ is a k-clique in G. A node of G as a subgraph of G is of course a 1-clique and an edge of G as a subgraph can be viewed as a 2-clique. A k-clique Δ is maximal if it cannot be extended to a $(k+1)$-clique in G by adding a further node of G to Δ. A k-clique Δ in G is a maximum clique if G does not contain any $(k+1)$-clique. A maximum clique in G is always maximal in G but a maximal clique in G is not necessarily a maximum clique in G. For each finite simple graph G there is a number k such that G contains a k-clique but G does not contain any $(k+1)$-clique. This well defined number k is called the clique number of G and it is denoted by $\omega(G)$.

Problem 1 *Given a finite simple graph* $G = (V, E)$. *Determine* $\omega(G)$.

Problem 2 *Given a finite simple graph* $G = (V, E)$ *and given a positive integer* k. *Decide if* G *contains a* k*-clique.*

Problem 3 *Given a finite simple graph* $G = (V, E)$ *list all maximum cliques that appear in* G.

Problem 4 *Given a finite simple graph* $G = (V, E)$ *list all maximal cliques that appear in* G.

Problem 1 is referred to as the maximum clique problem. It is an optimization problem and by the complexity theory of the algorithms it belongs to the NP-hard complexity class. (For further details see [2].)

Problem 2 is referred to as the k-clique problem. It is a decision problem and by the complexity theory of the algorithms it belongs to the NP-complete complexity class. (For further details see [4].) The four problems above all have important applications in discrete applied mathematics.

Some of the clique search problems are optimization problems and many of these algorithms have the following outline. Using computationally affordable techniques upper and lower bounds for the clique number of the given graph are established. If the lower and upper estimates agree, then the clique number of the graph is computed. If there is a gap between the upper and lower estimates, then we divide the clique search instance into smaller instances. In other words one carries out an optimality test and when this test is inconclusive a branching takes place.

Let $G = (V, E)$ be a finite simple graph and let P, Q, R be subsets of the set of nodes of G. The ordered triplet (P, Q, R) is called a splitting partition of the graph G if the following conditions are all satisfied.

(1) $P \cup Q \cup R = V$.

(2) $P \neq \emptyset$, $R \neq \emptyset$.

(3) $P \cap Q = P \cap R = Q \cap R = \emptyset$.

(4) $p \in P$, $r \in R$ implies that the unordered pair $\{p, r\}$ is not an edge of the graph G.

Let H be the subgraph of G induced by the set of nodes $P \cup Q$ and let K be the subgraph of G induced by the set of nodes $Q \cup R$. Let us suppose that Δ be a clique in G. In [6] it was proved that either Δ is a clique in H or Δ is a clique in K. This result is in an intimate relation with clique search procedures.

Let us suppose that we are looking for a maximum clique in the graph G. By the observation above we may restrict our attention to look for a maximum clique in the smaller graphs H and K. The larger are the sizes of the sets P and R the smaller are the subgraph H and K. Thus setting up a computationally economic branching rule in a maximum clique or in a k-clique algorithm depends on our ability to locate a splitting partition in a computationally economic manner.

As the main result of this paper we will propose a method to speedily locate splitting partitions in a given graph. The procedure we propose is rather myopic and so there is no any guarantee that the procedure provides splitting sets with optimal P and R sets. Unfortunately we do not possess theoretical tools to establish performance measurements of the splitting set spotting algorithm. We will carry out numerical experiments to demonstrate that the procedure works reasonably well.

2 Metric spaces and splitting partitions

Let $G = (V, E)$ be a finite simple graph and let u and v be two nodes of G. Set $d(u, v)$ to be the length of a shortest path leading from node u to node v. If there is no path from node u to node v we set $d(u, v)$ to be ∞. It may happen that there are more than one shortest paths leading from u to v. But their lengths must be the same. The quantity $d(u, v)$ can play the role of a distance between the nodes of G and the graph G can be viewed as a metric space equipped with this distance function.

For a vertex v of G the set of nodes adjacent to v is called the set neighbors of v and it is denoted by $N(v)$. In notation $N(v) = \{u : u \in V, \{v, u\} \in E\}$. The number of the elements of the set $N(v)$ is referred to as the degree of the node

v and it is denoted by $\deg(v)$. In a more general setting for a vertex v of G and for a subset U of V we define the degree of v with respect to the subset U as the number of neighbors of v in the subset U. We denote this restricted degree of v by $\deg_U(v)$. Plainly, $\deg_U(v) = |N(v) \cap U|$ and further $\deg(v) = \deg_G(v)$.

Set $\text{ball}_1(v) = \{v\} \cup N(v)$ and note that it is a ball of radius 1 centered at the point v in the metric space. For a subset U of V we define U^c to be the union of $\text{ball}_1(u)$ as u ranges over the elements of U. We may call the set U^c the closure of the set U. Condition (4) in the definition of the splitting partition can be expressed coveniently in terms of closure of the sets involved.

Lemma 5 *Let* $G = (V, E)$ *be a finite simple graph and let* P*,* Q*,* R *be subsets of* V *such that the ordered triplet* (P, Q, R) *is a splitting partition of* G*. Then*

$$P^c \cap R = \emptyset, \quad P \cap R^c = \emptyset \tag{1}$$

must hold.

Proof. Let us assume assume on the contrary that the ordered triplet (P, Q, R) is a splitting partition of G and in addition $P^c \cap R \neq \emptyset$ holds. In this situation there is a $p \in P$ and an $r \in R$ such that $r \in \text{ball}_1(p)$. It follows that the unordered pair $\{p, r\}$ is an edge of G. This contradicts condition (4) in the definition of the splitting partition. Assuming that $P \cap R^c \neq \emptyset$ a similar reasoning gives the contradiction again that the unordered pair $\{p, r\}$ is an edge of G. □

Lemma 6 *Let* $G = (V, E)$ *be a finite simple graph and let* P*,* R *be subsets of* V*. Suppose that beside condition (1) in Lemma 5 the condition*

$$P \neq \emptyset, \quad R \neq \emptyset \tag{2}$$

also holds. Then setting $Q = V \setminus (P \cup R)$ *the ordered triplet* (P, Q, R) *is a splitting partition of* G*.*

Proof. It is easy to see that each of the conditions (1), (3) in the definition of the splitting partition holds. Clearly, condition (2) in the definition of the splitting partition holds as a consequence of condition (2) in Lemma 6.

It remains to show that condition (4) in the definition of the splitting partition also holds. In order to do so assume on the contrary that condition (4) does not hold, that is, there is a $p \in P$ and an $r \in R$ such that the unordered pair $\{p, r\}$ is an edge of G. In this situation $r \in \text{ball}_1(p)$ and consequently $P \cap R^c \neq \emptyset$. This is in contradiction with the first part of condition (1) in Lemma 5. □

Lemma 7 *Let* $G = (V, E)$ *be a finite simple graph and let* P *be a subset of* V*. If* P *satisfies the condition*

$$P \neq \emptyset, \quad P^c \neq V. \tag{3}$$

Then setting $R = V \setminus P^c$ *for the sets* P *and* R *condition (1) in Lemma 5 and condition (2) in Lemma 6 hold.*

Proof. As $P \neq \emptyset$ holds by assumption, we need to prove only $R \neq \emptyset$ to get condition (2) in Lemma 6. But $R \neq \emptyset$ is a consequence of the assumption $P^c \neq V$.

The way the set R is constructed from P^c shows that the equation $P^c \cap R = \emptyset$ must hold. The equation $P \cap R^c = \emptyset$ follows from $P^c \cap R = \emptyset$. This gives that condition (1) in Lemma 5 is satisfied. $\square$

By Lemmas 5, 6, 7, constructing a splitting partition (P, Q, R) for G can be reduced to finding a subset P of V satisfying condition (3) in Lemma 7. This condition can be satisfied easily. For example the choice $P = \{v\}$ is a suitable choice whenever v is node of G that is not adjacent to at least one node of G. In this case $P = \{v\}$, $Q = N(v)$, $R = V \setminus (\{v\} \cup N(v))$. In fact, the splitting partition (P, Q, R) constructed in this way is the most commonly used branching rule in clique search algorithms. It is part of the Carraghan-Pardalos algorithm [1] and it is part of the Östergård algorithm [3]. A splitting partition (P, Q, R) for which either $|P| = 1$ or $|R| = 1$ is coming free of charge. From this reason we call such splitting partition of G a trivial splitting partition.

If (P, Q, R) is a splitting partition for G we may construct a new splitting partition (P', Q', R') for G. We set $U = Q \cup R$ and locate a node u of U for which $\deg_R(u)$ is a minimum. Then we move u from U to P and move the neighbors of u in R to Q to get the sets P', Q', R'. For the sake of simplicity we may use the initial setting $P = \emptyset$, $Q = \emptyset$, $R = V$ and construct new triplets (P, Q, R) while the condition $|P| < |R|$ holds.

3 Two small size examples

In order to illustrate the results presented so far we work out a small size example in details.

Example 8 *Let us consider the graph* $G = (V, E)$*. Here* $V = \{1, \ldots, 6\}$*. The adjacency matrix of* G *is depicted in Table 2. Figure 1 shows a geometric representation of* G*.*

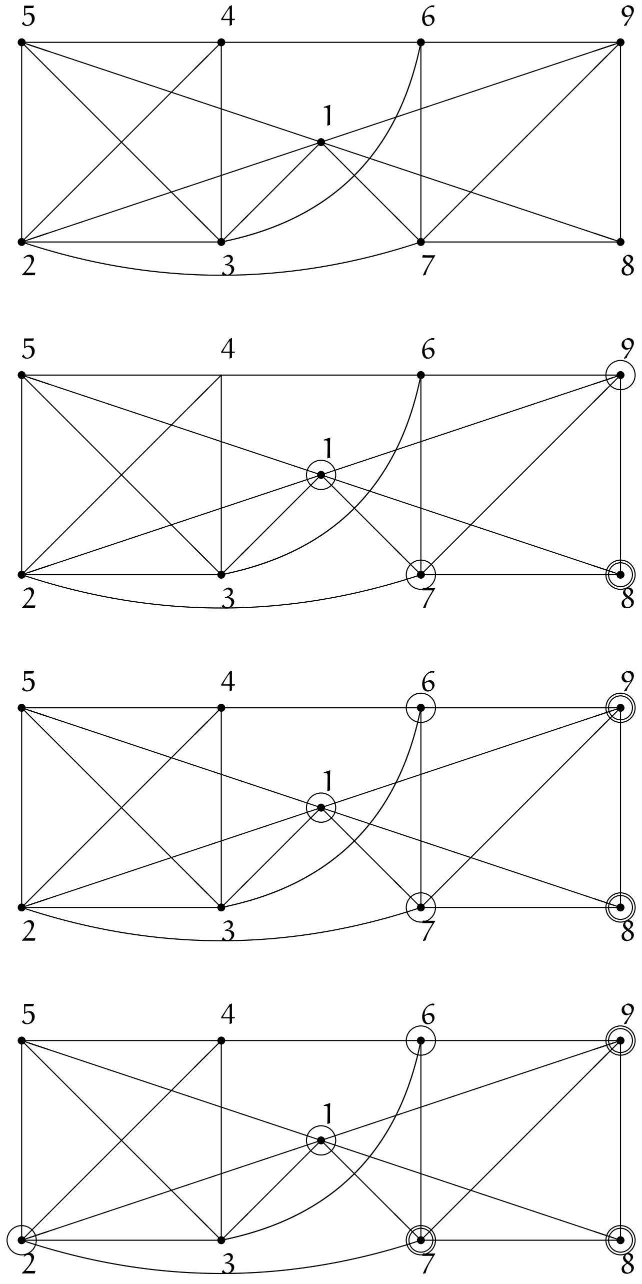


Figure 1: A graphical representation of the graph G in Example 8 and the steps of the procedure of spotting a splitting partition.

	1	2	3	4	5	6	7	8	9
1	×	●	●		●		●	●	●
2	●	×	●	●	●		●		
3	●	●	×	●	●	●			
4		●	●	×	●	●			
5	●	●	●	●	×				
6			●	●		×	●		●
7	●	●				●	×	●	●
8	●						●	×	●
9	●					●	●	●	×

	3	4	5	1	2	6	7	8	9
3	×	●	●	●	●	●			
4	●	×	●		●	●			
5	●	●	×	●	●				
1	●		●	×	●		●	●	●
2	●	●	●	●	×		●		
6	●	●				×	●		●
7				●	●	●	×	●	●
8				●			●	×	●
9				●		●	●	●	×

Table 1: The adjacency matrices of the graph in Example 8. In the second adjacency matrix we rearranged the rows and columns to make the splitting partition more apparent.

The reader can verify easily that the triplet (P, Q, R) of the subsets

$$P = \{3, 4, 5\}, \quad Q = \{1, 2, 6\}, \quad R = \{7, 8, 9\} \tag{4}$$

is a splitting partition of the graph G. Note that upper right and the lower left three by three submatrices are unfilled in the second adjacency matrix in Table 2.

We try to construct a splitting partition (P, Q, R) for the graph G. We set

$$P = \emptyset, \quad Q = \emptyset, \quad R = \{1, \ldots, 9\}.$$

The conditions (1), (3), (4) in the definition of splitting partition are satisfied. Condition (2) is not satisfied. We compute the degree of each node in $U = Q \cup R$ with respect to the set R.

node	1	2	3	4	5	6	7	8	9
degree	6	5	5	4	4	4	5	3	4

Node 8 has a minimum degree. We move node 8 from set R to set P. We move the neighbors of 8 from set R to set Q. In this way we get

$$P = \{8\}, \quad Q = \{1, 7, 9\}, \quad R = \{2, 3, 4, 5, 6\}.$$

The conditions (1), (2), (3), (4) in the definition of splitting partition are satisfied and consequently we have a genuine splitting partition of G. The

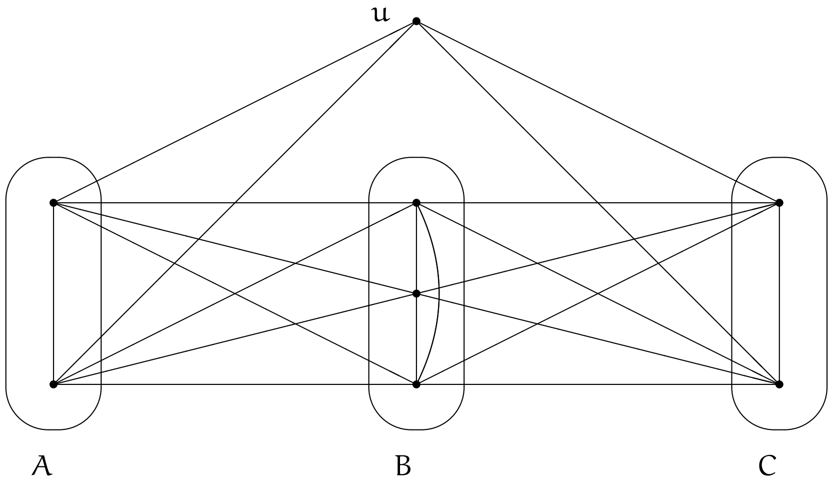


Figure 2: The graph G in Example 9.

sizes of the sets P and R are far from each other. We try to enlarge |P| even if this results a smaller |R|.

We compute the degree of each node in $U = Q \cup R$ with respect to the set R.

node	1	7	9	2	3	4	5	6
degree	3	2	1	3	4	4	3	2

Node 9 has a minimum degree. We move node 9 from set Q to set P. We move the neighbors of 9 from set R to set Q. In this way we get

$$P = \{8, 9\}, \quad Q = \{1, 7, 6\}, \quad R = \{2, 3, 4, 5\}.$$

The conditions (1), (2), (3), (4) in the definition of splitting partition are satisfied and consequently we have a splitting partition of G where the difference between |P| and |R| is reduced.

We compute the degree of each node in $U = Q \cup R$ with respect to the set R.

node	1	7	6	2	3	4	5
degree	2	1	1	3	3	4	3

Node 7 has a minimum degree. We move node 7 from set Q to set P. We move the neighbors of 7 from set R to set Q. In this way we get the splitting partition

$$P = \{7, 8, 9\}, \quad Q = \{1, 2, 6\}, \quad R = \{3, 4, 5\}$$

	1	2	3	4	5	6	7	8
1	×	●	●				●	●
2	●	×	●	●	●	●		
3	●	●	×	●	●	●		
4		●	●	×	●	●	●	●
5		●	●	●	×	●	●	●
6		●	●	●	●	×	●	●
7	●			●	●	●	×	●
8	●			●	●	●	●	×

Table 2: The adjacency matrix of the graph G in Example 9.

of G. This splitting partition is essentially the same as (4). The steps of the procedure can be followed on the geometric version of the graph G. Figure 1 shows these steps. The elements of the set R are marked with a double circle and the elements of the set Q are marked with a simple circle. Finally the elements of the set P are left unmarked.

We exhibit now an example to illustrate that the algorithm for spotting splitting partition described in the paper is a myopic one. Let A, B, C be pair-wise disjoint sets and let u be an element such that $u \notin (A \cup B \cup C)$. Let us assume that $|A| = |C| = n$ and $|B| = n + 1$. Using the sets A, B, C, $\{u\}$ we construct a graph $G = (V, E)$. We set $V = A \cup B \cup C \cup \{u\}$. We draw edges between nodes such that the subgraph induced by the set $A \cup B$ is a clique in G and similarly the subgraph induced by the set $B \cup C$ is a clique in G. Finally, we connect node u to each node in $A \cup C$. The reader can verify that with the $P = A$, $Q = B$, $R = C$ choices the ordered triplet (P, Q, R) is a splitting partition of G. Here $|P| = n$ and $|R| = n$. On the other hand, the greedy algorithm proposed by the paper will locate the splitting partition (P, Q, R), where $P = \{u\}$, $Q = A \cup C$, $R = B$. Here $|P| = 1$ and $|R| = n + 1$. We can see that for $n \geq 2$ the graph G has a non-trivial splitting partition. But the greedy algorithm locates a trivial splitting partition. The $n = 2$ particular case of the above construction is the content of the next example.

Example 9 *Set* $A = \{2, 3\}$, $B = \{4, 5, 6\}$, $C = \{7, 8\}$, $u = 1$. *Let us consider the graph* $G = (V, E)$, *where* $V = \{1, \ldots, 8\}$. *The adjacency matrix of* G *is depicted in Table 3. Figure 2 shows a possible geometric representation of* G.

4 Numerical experiments

For testing purposes we have selected three infinite families of graphs that are connected to the existence and construction of certain error detecting and error correcting codes. The so-called monotonic matrices are in intimate connection with codes over the alphabet $\{1, \ldots, n\}$. Each code words has length three. The problem is to find a code whose inner distance is at least two. (See [6], [8].) The deletion error detecting codes are consisting of binary code words of length n. These words are sent over a noisy channel. Due to transmission error on the receiver side a shorter word may arrive. The task is to devise a code that makes possible to detect a one bit deletion error. (For further details see [5].) The Johnson codes we are considering here are binary codes with word length n. Each code word consists of 4 1's and $n - 4$ 0's. The Hamming distance of two distinct code words is at least 3.

Monoton				Deletion				Johnson			
n	$\|V\|$	α	β	n	$\|V\|$	α	β	n	$\|V\|$	α	β
3	27	4	4	3	8	2	4				
4	64	5	7	4	16	4	4				
5	125	7	8	5	32	4	5				
6	216	9	9	6	64	4	5	6	15	2	4
7	343	10	12	7	128	4	7	7	35	2	5
8	512	12	13	8	256	5	5	8	70	2	6
9	729	14	14	9	512	5	8	9	126	3	3
10	1 000	15	17	10	1024	6	6	10	210	3	4
11	1 331	17	18	11	2048	6	10	11	330	4	4
12	1 728	19	19	12	4096	7	7	12	495	4	5
13	2 197	20	22					13	715	5	5
14	2 744	22	23					14	1 001	5	6
15	3 375	24	24					15	1 365	6	6
16	4 096	25	27					16	1 820	6	7
17	4 913	27	28					17	2 380	7	7
								18	3 060	7	8
								19	3 876	8	8
								20	4 845	8	9

Table 3: Numerical results in connection with graphs coming from coding theory.

The results of the numerical experiments are summarized in the Table 3. We describe the meaning of the entries using the 10-th row of Table 3 as an illustration. A graph G is associated with a monotonic matrix of parameter $n = 10$. The graph has $|V| = 1000$ vertices. These values are in the first two columns of the table. The splitting partition (P, Q, R) we have spotted has the parameters $|P| = \alpha = 15$, $|R| = \beta = 17$ and the next two columns contain these α, β values.

At this stage we may conclude that the algorithm spots splitting partitions rapidly and works reliably in connection with non-trivial size graphs. Only after working with the algorithm for a longer period of time involving a much wider variety and range of graphs would enable us to assess the merits of the proposed procedure.

References

[1] R. Carraghan, P. M. Pardalos, An exact algorithm for the maximum clique problem, *Operation Research Letters* **9** (1990), 375–382. ⇒135

[2] M. R. Garey, D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman, New York, 2003. ⇒132

[3] P. R. J. Östergård, A fast algorithm for the maximum clique problem, *Discrete Applied Mathematics* **120** (2002), 197–207. ⇒135

[4] C. H. Papadimitriou, *Computational Complexity*, Addison-Wesley Publishing Company, Inc., Reading, MA 1994. ⇒132

[5] N. J. A. Sloane, Challenge Problems: Independent sets in graphs, `http://neilsloane.com/doc/graphs.html` ⇒140

[6] S. Szabó, Parallel algorithms for finding cliques in a graph,, *Journal of Physics, Conference Series* **268** (2011) 012030 DOI:10.1088/1742-6596/268/1/012030. ⇒ 131, 133, 140

[7] S. Szabó, Monoton matrices and finding cliques in a graph, *Annales Univ. Sci. Budapest., Sect. Computatorica* **41** (2013), 307–322. ⇒

[8] E. W. Weisstein, Monotonic Matrix, In: *MathWorld–A Wolfram Web Resource.* `http://mathworld.wolfram.com/MonotonicMatrix.html` ⇒140

Received: September 26, 2019 • Revised: October 19, 2019

Acta Univ. Sapientiae, Informatica **11,** 2 (2019) 142–158

DOI: 10.2478/ausi-2019-0010

Evolutionary solving of the debts' clearing problem

Csaba PĂTCAȘ
Babeș-Bolyai University
Cluj-Napoca, Romania
email: patcas@cs.ubbcluj.ro

Attila BARTHA
Babeș-Bolyai University
Cluj-Napoca, Romania
email: abartha@yahoo.com

Abstract. The debts' clearing problem is about clearing all the debts in a group of n entities (banks, companies etc.) using a minimal number of money transaction operations. The problem is known to be NP-hard in the strong sense. As for many intractable problems, techniques from the field of artificial intelligence are useful in finding solutions close to optimum for large inputs. An evolutionary algorithm for solving the debts' clearing problem is proposed.

1 Introduction

The problem of debt clearing (DC problem) can arise among a group of friends, but it also needs to be solved regularly among the affiliates of multinational corporations, banks or even countries ([16, 18]). As money transactions are time- and money-sensitive operations, it can be desirable to clear the debts in a minimal number of money transaction operations.

Problems related to debt clearing were studied in the past. Shapiro gave a linear programming based model to minimize the cost of payments netting assuming that costs are directly proportional to the volume transacted ([16]).

Computing Classification System 1998: I.2.8
Mathematics Subject Classification 2010: 68T20
Key words and phrases: debt settling, genetic algorithm

In [18] a network flow based model was given which solves some of the weaknesses of Shapiro's model and is more efficient computationally. Because in our problem the goal is to minimize the number of transactions in a multilateral netting system, there is no linearity of costs, thus neither of these methods can be used as it was briefly discussed in [13].

In [9] the NP-complete Bank Clearing Problem (BCP) was introduced as it occurred in Germany's largest interbank payment system and efficient heuristic algorithms were given to solve it. Later in [17] an approximation algorithm for the BCP was given. In the BCP the objective is to maximize the clearing volume with the restriction that the negative net balance cannot exceed a previously deposited amount for each bank. Because the objective and the constraints of the BCP are different from the version of the DC problem discussed here (where the objective is to minimize the number of transactions), these heuristic algorithms cannot be adapted to solve the DC problem and cannot provide a comparison for our proposed evolutionary algorithm.

In [11] a survey is given on solving some other banking related issues (such as portfolio optimization, bankruptcy prediction and FOREX rate prediction) using evolutionary computing. A stochastic model for a payment and settlement system capable of processing payments in real time is presented in [1] by the example of the Clearinghouse of the Bank of Lithuania. Using this model, in [2] several FIFO clearing algorithms are tested by simulation.

The problem of mutual debts compensation (MDC) is formulated using graph theory in [4, 5]. The author proposes a cycle elimination method, but also shows by an example that the order of elimination is important, which is also mentioned in [13]. In this problem the goal is to maximize the total amount of eliminated debts and can be solved efficiently by linear programming and also by network flow methods. In [6] new models for MDC are introduced.

The problem of settling debts in a minimal number of transactions was discussed by Verhoeff in 2004 ([21]).

Pătcaş [13] later re-discovered the problem and proposed it in 2008 at the qualification contest of the Romanian national team of informatics. The solution was described in [13] and the problem conjectured to be intractable, which was earlier proved in [21]. In [15] the problem's relation to complexity classes was further studied. In [14] the problem in a dynamic setting is discussed and a new algorithm given, having superior speed in some cases compared to the one described in [13].

List of borrowings:

Borrower	Lender	Amount of money
1	3	4
3	4	7
4	2	2
2	1	2
1	5	1
3	5	1
5	4	2

Solution:

Sender	Reciever	Amount of money
1	4	3
3	4	4

Figure 1: Example for the DC problem

2 Stating the problem

The problem statement is the following([13]):

Let us consider a number of n *entities (persons, companies etc.), and a list of* m *borrowings among these entities. A borrowing can be described by three parameters: the index of the borrower entity, the index of the lender entity and the amount of money that was lent. The task is to find a minimal list of money transactions that clears the debts formed among these* n *entities as a result of the* m *borrowings made.*

It is natural to model this problem using graph theory. Consider the following definitions.

Definition 1 ([13]) *Let* $G(V, A, W)$ *be a directed, weighted multigraph without loops,* $|V| = n$, $|A| = m$, $W : A \to \mathbb{Z}$, *where* V *is the set of vertices,* A *is the set of arcs and* W *is the weight function.* G *represents the borrowings made, so we will call it the* ***borrowing graph****.*

The borrowing graph corresponding to the example in Figure 1 is depicted in Figure 2.

Definition 2 ([13]) *Let us define for each vertex* $v \in V$ *the* ***absolute amount of debt*** *over the graph* G*:* $D_G(v) = \sum_{\substack{v' \in V \\ (v,v') \in A}} W(v, v') - \sum_{\substack{v'' \in V \\ (v'',v) \in A}} W(v'', v)$

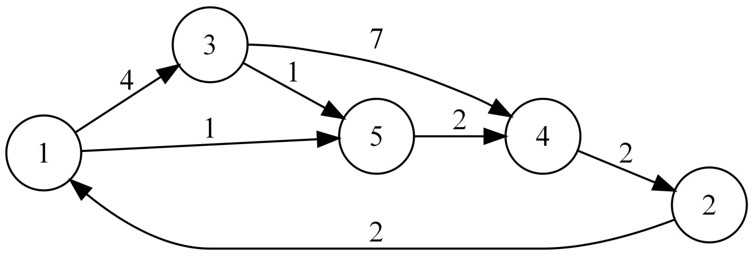


Figure 2: The borrowing graph associated with the given example. An arc from node i to node j with weight w means, that entity i must pay w amount of money to entity j.

i	1	2	3	4	5
$D(i)$	3	0	4	-7	0

Figure 3: Absolute amounts of debt corresponding to the given example.

Sometimes for simplicity we will refer to the absolute amount of debt of a node as D***-value****.*

The D-values corresponding to the example from Figure 1 are listed in Figure 3.

Definition 3 ([13]) *Let* $G'(V, A', W')$ *be a directed, weighted multigraph without loops, with each arc* (i, j) *representing a transaction of* $W'(i, j)$ *amount of money from entity* i *to entity* j*. We call this graph a* ***transaction graph****. These transactions clear*[1] *the debts formed by the borrowings modeled by graph* $G(V, A, W)$ *if and only if:*

$D_G(v_i) = D_{G'}(v_i), \forall i = \overline{1, n}$*, where* $V = \{v_1, v_2, \ldots, v_n\}$

We will note this by: $G \sim G'$*.*

See Figure 4 for a transaction graph with minimal number of arcs corresponding to the example from Figure 1.

We are now ready to reformulate the problem mathematically:

[1]When trying to decide if the transactions described by a transaction graph clear the debts represented by a borrowing graph, it is easy to see that only D-values matter ([21]).

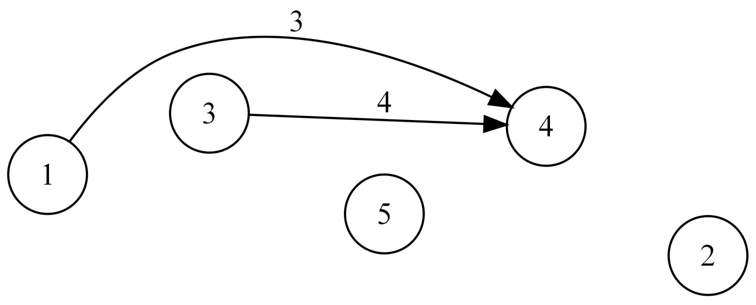


Figure 4: The respective minimum transaction graph. An arc from node i to node j with weight w means, that entity i pays w amount of money to entity j.

Given a borrowing graph $G(V, A, W)$ *we are looking for a minimal transaction graph* $G_{min}(V, A_{min}, W_{min})$*, so that* $G \sim G_{min}$ *and* $\forall G'(V, A', W') : G \sim G', |A_{min}| \leq |A'|$ *holds.*

3 An equivalent problem

The following observation is crucial in all of the solutions known so far.

Theorem 4 ([21, 13]) *Any instance of the DC problem can be solved trivially by at most* $n - 1$ *transactions.*

Proof. We give an algorithmic proof.

1. Let us choose two nodes i and j, such that $D(i) > 0$ and $D(j) < 0$.
2. Add arc (i, j) to the transaction graph having weight $\min(D(i), -D(j))$.
3. Update the D-values of i and j to reflect the addition of the arc (by decreasing $D(i)$ and increasing $D(j)$).
4. Repeat steps (1) - (3) as long as possible.

It is clear that at least one D-value becomes zero as a result of executing steps (1) - (3). Also, because we have the invariant that the sum of all D-values is always zero, at the last iteration we always have $D(i) = -D(j)$. Thus two D-values become zero at the last iteration, which yields the needed upper bound. □

We observe that finding a minimal transaction graph is equivalent to partitioning V into a maximal number of disjoint zero-sum subsets, more formally $V = P_1 \cup \ldots \cup P_{max}$, $\sum\limits_{u \in P_i} D(u) = 0, \forall i = \overline{1, max}$ and $P_i \cap P_j = \emptyset, \forall i, j = \overline{1, max}, i \neq j$. The reason for this is, that all the debts in a zero-sum subset P_i can be cleared by $|P_i| - 1$ transactions by Theorem 4, thus to clear all the debts, $|V| - max$ transactions are necessary.

4 Evolutionary technique for solving the DC problem

We use the reformulation of the problem described in Section 3.

Representation In our method we represent a solution of the problem by a permutation of the D-values of V, the set of nodes. Thus a candidate solution is a vector $C = (c_1, c_2, \ldots, c_n)$, such that $c_i = D(u), \forall i \in \overline{1, n}$ for some unique $u \in V$.

For instance $C = (3, 0, -7, 4, 0)$ is a chromosome representing a candidate solution for the D-values from Figure 3.

The idea of representing solutions as permutations was discussed intensively in the context of the Traveling Salesman Problem (TSP) ([7, 12, 22]).

Fitness assignment To evaluate the fitness of a chromosome, we need to determine the number of zero-sum subsets codified by the candidate solution, taking into consideration the order of appearance of the D-values. In order to calculate it, we iterate over the genes of the chromosome from left to right and maintain the partial sum obtained so far, that is $s_i = \sum\limits_{j=1}^{i} c_j$. For every $s_i = 0$, we have found a new zero-sum subset of the partition (starting after the last encountered partial sum equal to zero and ending at i), so we can add one to the fitness of the chromosome.

For instance if we have $C = (-3, 2, 1, -5, 5)$, then $s = (-3, -1, 0, -5, 0)$, so the fitness of C will be 2, corresponding to the partition formed by the first three elements and the last two elements.

Recombination Various operators for permutation representations are discussed in [3, 7, 8, 12, 19, 20, 22, 23].

When selecting existing recombination operators or designing new ones the representation of the problem is crucial to consider. In our case we had to take into account that we have a permutation representation corresponding to disjoint sets whose number has to be maximized. Thus for our problem the resulting offsprings of a crossover must represent a valid permutation and must have the potential to improve the number of zero-sum sets.

Because these sets are constructed by looking at the order of genes, for operator `Recomb1` we have chosen an order-based crossover method, in particular the Modified Order Crossover (MOX) operator described in [23]. Most crossover operators for permutation representations are designed for the TSP. Replacing a few arcs in the solution of the TSP usually does not greatly change the fitness of the solution, but may have a negative impact in the DC problem by perturbing too many zero-sum sets. Thus we had to be careful which operators to adapt to our problem.

For operator `Recomb2` we propose a new crossover operator, which intuitively has a great potential in increasing the number of subsets codified by the offsprings, by leveraging information from the parents.

Recomb1 Let C_1 and C_2 be the two chromosomes, and $k \in [1, n]$ a random crossover point. Then, the first descendant C_1' can be obtained by copying the first k genes from C_1 and appending to it the elements of the permutation not used so far in the same order as they appear in C_2. The second descendant C_2' is obtained symmetrically.

For instance,

$$\begin{gathered} k = 2 \\ C_1 = (\mathbf{-3}, \mathbf{2}, 1, -5, 5)\ C_2 = (\mathbf{-5}, \mathbf{2}, 1, -3, 5) \\ \downarrow \\ C_1' = (\mathbf{-3}, \mathbf{2}, -5, 1, 5)\ C_2' = (\mathbf{-5}, \mathbf{2}, -3, 1, 5) \end{gathered}$$

Recomb2 The problem with `Recomb1` is, that the first descendant inherits most of its properties from C_1 and very little from C_2. Symmetrically C_2' inherits most of its properties from C_2 and very little from C_1. This is undesirable, as both C_1 and C_2 can contain subsets from the optimal partition.

A better recombination operator may be the following. First, determine the zero-sum subsets codified by C_1 and C_2, as described at the fitness assignment.

Let those be $C_1 = P_{1,1} \cup P_{1,2} \cup \dots$ and $C_2 = P_{2,1} \cup P_{2,2} \cup \dots$. Initialize $C_1' := C_1$ and $C_2' := C_2$.

Then, iterate over every $P_{1,i}$. If some $P_{1,i}$ is contained in some $P_{2,j}$, that is $P_{1,i} \subset P_{2,j}$, replace $P_{2,j}$ in the second descendant with $P_{1,i} \cup (P_{2,j} \setminus P_{1,i})$[2] Repeat the same procedure for C_2 symmetrically.

For instance,

$$\begin{gathered} C_1 = (-3, 2, 1, -5, 5) = \{-3, 2, 1\} \cup \{-5, 5\} \\ C_2 = (2, 1, 5, -5, -3) = \{2, 1, 5, -5, -3\} \\ \downarrow \\ C_1' = \{-3, 2, 1\} \cup \{-5, 5\} = (-3, 2, 1, -5, 5) \\ C_2' = \{-3, 2, 1\} \cup \{5, -5\} = (-3, 2, 1, 5, -5) \end{gathered}$$

Mutation In our experiments we have used three mutation operators. `Mut1` is a classical inversion operator.

We propose two new mutation operators with the property, that the fitness value of the chromosome does not decrease. The new mutation operators are based on `Mut1`, but are using the additional information of the chosen representation of our particular problem.

Mut1 Holland described an inversion operator in [10], which reverses the order of the elements between two randomly chosen indices. This method can be used without modification, on the sequence between the i^{th} and j^{th} elements.

For instance,

$$\begin{gathered} i = 2, j = 5 \\ C = (-3, \mathbf{2}, \mathbf{1}, -\mathbf{5}, \mathbf{5}) \\ \downarrow \\ C' = (-3, \mathbf{5}, -\mathbf{5}, \mathbf{1}, \mathbf{2}) \end{gathered}$$

Mut2 `Mut1` can be used on the partition $C = P_1 \cup P_2 \cup \dots$ instead of the permutation representation. This method guarantees that the fitness of the chromosome does not decrease.

For instance,

[2]In our implementation we have chosen to put the elements of $P_{1,i}$ into C_2' in the same order as they were in C_1 and the elements of $P_{2,j} \setminus P_{1,i}$ in the same order as they were in C_2. Other variations are possible as well.

$$\begin{gathered} i = 1, j = 4 \\ C = (-2, 2, 3, 4, -7, 1, -1, 6, -3, 2, -5) = \\ \{\mathbf{-2, 2}\} \cup \{\mathbf{3, 4, -7}\} \cup \{\mathbf{1, -1}\} \cup \{\mathbf{6, -3, 2, -5}\} \\ \downarrow \\ C' = \{\mathbf{6, -3, 2, -5}\} \cup \{\mathbf{1, -1}\} \cup \{\mathbf{3, 4, -7}\} \cup \{\mathbf{-2, 2}\} = \\ (6, -3, 2, -5, 1, -1, 3, 4, -7, -2, 2) \end{gathered}$$

Mut3 `Mut1` can also be used inside some P_k without decreasing the fitness. For instance,

$$\begin{gathered} k = 4, i = 1, j = 4 \\ C = (-2, 2, 3, 4, -7, 1, -1, 6, -3, 2, -5) = \\ \{-2, 2\} \cup \{3, 4, -7\} \cup \{1, -1\} \cup \{\mathbf{6, -3, 2, -5}\} \\ \downarrow \\ C' = \{-2, 2\} \cup \{3, 4, -7\} \cup \{1, -1\} \cup \{\mathbf{-5, 2, -3, 6}\} = \\ (-2, 2, 3, 4, -7, 1, -1, -5, 2, -3, 6) \end{gathered}$$

5 How to obtain large instances of the DC problem

Because our problem is NP-hard as demonstrated in [15], it is challenging to generate large test cases for which information about the optimal solution is known. We describe five methods to generate large test cases.

Method 1 If the optimal solution for some input is known, padding the set of D-values with k zeros increases the optimal solution also by k.

Method 2 Method 1 can be modified by padding the input with k pairs of the structure $(x, -x)$.

Method 3 If the number of negative (or positive) numbers is two, the problem is equivalent to the Subset Sum problem and is solvable in pseudopolynomial time by dynamic programming. Using this method we can generate inputs for which the optimal solution is unique, that is, there is a single subset of positive (negative) numbers having the sum equal to one of the two negative (positive) numbers (in absolute value). An optimal answer for such an input is expected to be difficult to find for our evolutionary approach, as in the worst case (when the cardinality of the subset is $n/2$) only $2 \cdot (\frac{n}{2}!)^2$ out of the $n!$ possible permutations do represent an optimal solution. For $n = 10$, this means that the ratio of optimal solutions and all solutions is about $7.9 \cdot 10^{-3}$, while for $n = 100$ the ratio is about $1.9 \cdot 10^{-29}$.

This idea can be extended for any fixed number of negative (positive) numbers, but the running time of the dynamic programming solution raises quickly.

Method 4 Let n be the desired size of the input and $l \leq \lfloor n/2 \rfloor$ an integer. First generate randomly a set of $n - l$ elements, containing only positive D-values and l distinct integers from the $[1, n - l]$ range (denoted $r_1 < \ldots < r_l$). Let s be the vector of partial sums, that is $s_i = \sum_{j=1}^{i} D(j), \forall i = \overline{1, n - l}$ (we assume $s_0 = 0$ and $r_0 = 0$). For every $r_i, \forall i = \overline{1, l}$ insert $-(s_{r_i} - s_{r_{i-1}})$ to the set. In other words we insert with a negative sign the sum of l partial sequences, whose borders are denoted by r_{i-1} and r_i. By this method we can get the optimal solution to be equal to l. As the range of the possible values of the first $n - l$ positive elements gets bigger, we expect the optimal solution to be harder and harder to find. The reason is that the probability to get the same sum from a different combination of positive numbers gets smaller, thus the number of genetic representations corresponding to an optimal solution decreases.

Method 5 It can be easily seen, that if the optimal solution for a set V is known to be max, then the solution for $V \cup V$ will be $2 \cdot max$, the solution for $V \cup V \cup V$ will be $3 \cdot max$ and so on.

6 Numerical experiments

A preliminary testing phase was carried out using the same 15 test cases which were used when the problem was proposed in 2008 at the qualification contest of the Romanian national team (see [14] for a description of each instance). These test cases all have specially crafted structures, with $n \leq 20$, $m \leq 100$ and the cost of an arc being a natural number no larger than 100. For comparison, the optimal solution was determined for each test case by using the algorithm described in [13].

Because these instances have small size, our genetic algorithm can be used with a wide range of parameters and operators to reach the optimal solution in a matter of seconds.

To test the above statement empirically, we used a population of 100 individuals and the number of generations was set 100. We used operator `Recomb2` in conjunction with roulette wheel selection and operator `Mut1` with a mutation probability of 0.5. The best five individuals always survived to the next generation. Our genetic algorithm found the best solution for all of the test cases.

6.1 Combinations of operators

In the first set of experiments our goal was to determine which combinations of our recombination and mutation operators work best in practice, along with desirable values for mutation probability. We constructed three test cases (`debt100a`, `debt100b` and `debt100c`)[3] with different structures, all of them having $n = 100$

`debt100a` was obtained by concatenating the test case from the initial 15 which was the most difficult to solve for the genetic algorithm (case 15) five times to itself. By the observation above in Method 5, the optimal solution for this test case is $max = 25$.

To generate `debt100b` we used Method 3 for $n = 50$ and concatenated the obtained set once to itself, thus obtaining a case having $max = 4$ by the observation above.

To obtain `debt100c` we first generated, using a dynamic programming algorithm, a set having 20 elements, which can be uniquely partitioned into three zero-sum subsets (and no more). Then we concatenated this set five times to itself, yielding $max = 15$ for this test case.

For each of the three described test cases we used the following methodology. For every possible combination of recombination and mutation operators we fixed the mutation probability to every value from 0 to 1 in steps of 0.1 and executed the genetic algorithm 10 times. We recorded the best solution obtained among the 10 executions, the average of the 10 best values and the average fitness of all genomes. In each case the population size was set to 100 individuals and the number of generations to 1000. For the recombination operators roulette wheel selection was used in every case and the best five individuals always survived to the next generation.

To assess the efficacy of our algorithm we compared it to an algorithm called `RandomSearch`, which works by generating an independent random solution in every generation for each chromosome. In our case this meant generating 100000 random solutions and remembering the one with the maximum fitness value among them.

[3]All test cases used in our experiments can be downloaded from `http://cs.ubbcluj.ro/~patcas/debt_experiments.zip`

The results of the first set of experiments were the following:

- `debt100c` was the most difficult of the three test cases used, no algorithm being able to find the optimal solution $\mathtt{max} = 15$. The best solution found by `RandomSearch` was 5, and the best solution found by the evolutionary algorithms was 13, using `Recomb2` along with `Mut1` with a mutation probability ranging from 0.8 to 1. The average fitness of all genomes was maximal at mutation probability 0.7.
- `debt100b` was the easiest of the test cases, our genetic algorithm being able to find the optimal solution $\mathtt{max} = 4$ in the majority of the cases (in about 76% of the possible combinations of recombination and mutation operators and mutation probabilities). Mutation probability 0.7 along with `Recomb2` and `Mut1` maximized the average fitness again. The best solution found by `RandomSearch` was 3.
- For `debt100a` `RandomSearch` was able to find a solution with fitness 9. Our genetic algorithm found the optimal solution 25 in a small percentage of the cases, using the same parameters that yielded the best solutions for `debt100c`. Maximal average fitness was obtained with mutation probability 0.4 using `Recomb2` and `Mut1`.

We can draw the conclusion that our genetic algorithm is much more efficient than generating random solutions. The results suggest that using `Recomb2` with `Mut1` works best in practice for a wide range of inputs. On the other hand we note that `Recomb2` and `Mut2` is a particularly bad combination, the reason being that it does not allow the exploration of a sufficient varied range of solutions, because neither of the operators is able to introduce new partition sets into the population. Still, `Mut2` works fairly well together with `Recomb1`, as the latter is capable of constructing new partition sets.

6.2 Convergence to optimum

In the second set of experiments we studied the convergence of the solution to the optimal value as the number of generations increases. We concatenated each of the three test cases described above ten times to itself, obtaining cases `debt1000a`, `debt1000b` and `debt1000c` respectively. We executed our genetic algorithm using `Recomb2` and `Mut1` with a mutation probability of 0.75. The population size was set to 80 and the best five individuals were always promoted to the next generation. The algorithm was executed once for

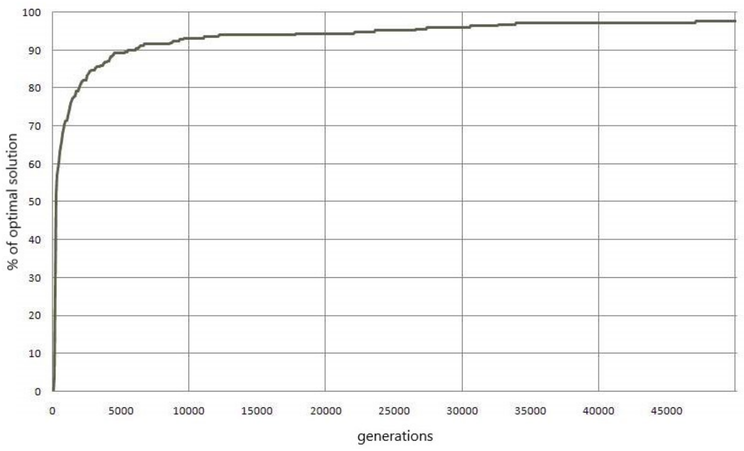


Figure 5: The fitness of the best individual compared to the optimal solution in percentages for test case `debt1000a` as the number of generations increases.

50000 generations, and the fitness of the best chromosome was recorded every 100 generations.

The results are depicted in Figures 5, 6 and 7. We can observe that in every case the fitness of the best individual raises sharply in the first 5000 generations, then slows down gradually. 50000 generations were enough to find a solution having fitness 244 (97.6% of the optimum) for `debt1000a` and a solution having fitness 39 (97.5% of the optimum) for `debt1000b`. Case `debt1000c` was significantly more difficult, the best solution having only fitness 122 (81.3% of the optimum).

6.3 Efficiency on very difficult test cases

In the third set of experiments we used Method 2 to generate test cases which are very difficult for our evolutionary algorithm. Starting with $n = 100$ and going by increments of 100 we generated sets having the structure $\{1, 2, \ldots, n/2, -1, -2, \ldots, -n/2\}$. It can be easily seen that the optimal solution for these cases is $max = n/2$ and it is unique. Only $\frac{n}{2}! \cdot 2^{n/2}$ representations out of $n!$ translate to an optimal solution, which means that the ratio of optimal solutions to all solutions is about $1.0 \cdot 10^{-3}$ for $n = 10$ and about $3.6 \cdot 10^{-79}$ for $n = 100$.

For every case we executed the genetic algorithm 10 times using `Recomb2`

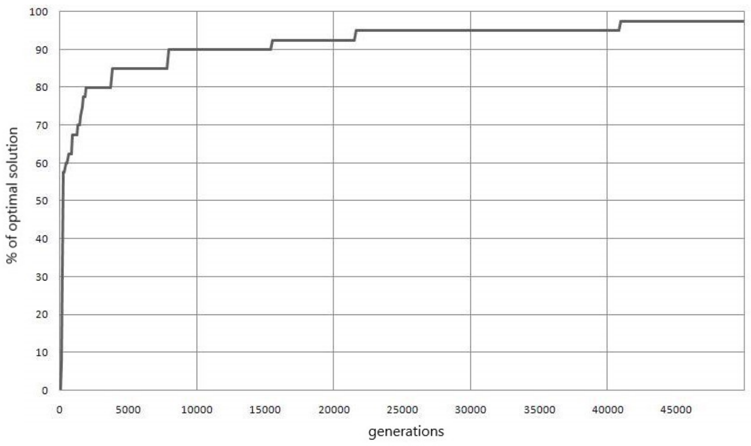


Figure 6: The fitness of the best individual compared to the optimal solution in percentages for test case `debt1000b` as the number of generations increases.

and `Mut1` with a mutation probability 0.75. The population size was set to 80 and the best five individuals were always promoted to the next generation. The algorithm was stopped after 5000 generations. For every test case we recorded the best solution found by the algorithm, the average of the best solutions over the 10 executions and the summed up running time of the 10 executions. The results are presented in Figure 8.

For $n = 100$ the optimal solution was found in all of the 10 executions, but as the size of the input increased, the best solution got further and further from the optimum. We note the robustness of the algorithm, as the best solution is usually just a few percentages away from the average.

7 Conclusions

The debts' clearing problem is an NP-hard problem of practical interest, as it arises in real life situations as well. The only known algorithms to solve the problem were the ones presented in [13] and [14], which are exact algorithms that always provide the optimal solution, but their running time is practical only for small inputs ($n \leq 20$).

Using an equivalent problem, we described an evolutionary algorithm to solve the problem and made extensive experiments to assess its efficacy. From the experiments we concluded that our algorithm is much more efficient than a random search in the space of the solutions. Our algorithm is capable of finding

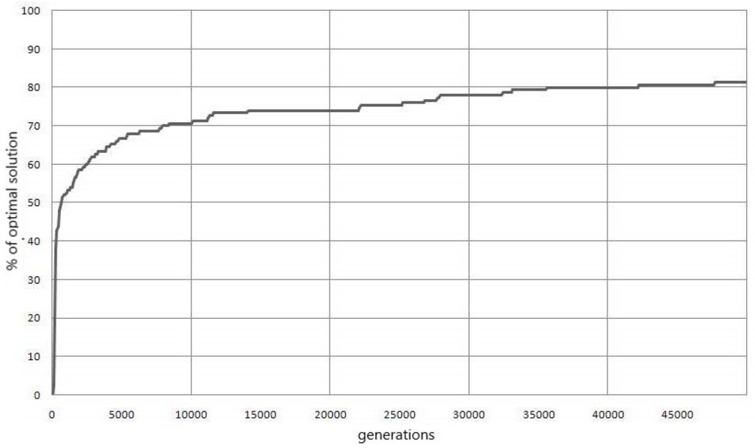


Figure 7: The fitness of the best individual compared to the optimal solution in percentages for test case `debt1000c` as the number of generations increases.

N	Best solution (% of optimum)	Average of bests (% of optimum)	Running time (in seconds)
100	50 (100%)	50 (100%)	203
200	76 (76%)	70.4 (70.4%)	710
300	91 (60.6%)	85.5 (57%)	1247
400	108 (54%)	100.5 (50.2%)	1919
500	116 (46.4%)	109.8 (43.9%)	2610
600	130 (43.3%)	121.1 (40.3%)	3328
700	138 (39.4%)	132 (37.7%)	4225
800	147 (36.7%)	142.4 (35.6%)	5134
900	155 (34.4%)	146.6 (32.5%)	6084
1000	166 (33.2%)	157.3 (31.4%)	6766

Figure 8: Results of 10 executions for 5000 generations each, on very difficult test cases

the optimal solution for the most difficult test cases with sizes up to $n = 100$ in a matter of minutes. For cases as large as $n = 1000$ our approach remains practical, as it can obtain solutions in the range of 80% - 98% compared to the optimal solution in about an hour on a personal computer. In comparison a random search does not go above 15% even for the easiest cases of this size.

References

[1] D. Bakšys and L. Sakalauskas, Modelling, simulation and optimisation of interbank settlements, *Information technology and control* **36,** 1 (2007) 43–52. ⇒143

[2] D. Bakšys and L. Sakalauskas, Simulation and testing of FIFO clearing algorithms, *Information technology and control* **39,** 1 (2010) 24–31. ⇒ 143

[3] L. Davis, Applying adaptive algorithms to epistatic domains, *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, 162–164, Morgan Kaufmann, 1985. ⇒148

[4] V. Gazda, Mutual debts compensation as graph theory application, *Challenges for Business Administrators in the New Millennium*, 793–811, 2000. ⇒143

[5] V. Gazda, Mutual debts compensation as graph theory problem, *Mathematical Finance*, 162–167, Springer, 2001. ⇒143

[6] V. Gazda, D. Horváth and M. Rešovskỳ, An application of graph theory in the process of mutual debt compensation, *Acta Polytechnica Hungarica* **12,** 3 (2015) 7–24. ⇒143

[7] D. Goldberg and R. Lingle, Jr. Alleles, loci, and the traveling salesman problem, *Proceedings of the 1st International Conference on Genetic Algorithms and their Applications*, 154–159, Lawrence Erlbaum Associates, 1985. ⇒147, 148

[8] M. Gorges-Schleuter, *Genetic algorithms and population structure - A massively parallel algorithm*, Ph.D. thesis, University of Dortmund, 1990. ⇒148

[9] M. M. Güntzer, D. Jungnickel and M. Leclerc, Efficient algorithms for the clearing of interbank payments, *European Journal of Operational Research* **106,** 1 (1998) 212–219. ⇒143

[10] J. H. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, 1975. ⇒149

[11] G. J. Krishna and V. Ravi, Evolutionary computing applied to solve some operational issues in banks, *Optimization in Industry* 31–53, Springer, 2019. ⇒143

[12] I. Oliver, D. Smith and J. Holland, A study of permutation crossover operators on the traveling salesman problem, *Proceedings of the Second International Conference on Genetic Algorithms*, 224–230, Lawrence Erlbaum Associates, 1987. ⇒147, 148

[13] C. Pătcaş, On the debts' clearing problem, *Studia Universitatis Babeş-Bolyai Series Informatica*, **54,** 2 (2009) 109–120. ⇒143, 144, 145, 146, 151, 155

[14] C. Pătcaş, The debts' clearing problem: a new approach, *Acta Universitatis Sapientiae, Informatica*, 3, 2 (2011) 192–204. ⇒143, 151, 155

[15] C. Pătcaş, The debts' clearing problem's relation with complexity classes, *Acta Mathematica Academiae Paedagogicae Nyíregyháziensis*, 28, 2 (2012) 217–226. ⇒143, 150

[16] A. C. Shapiro, Payments netting in international cash management, *Journal of International Business Studies*, 9, 2 (1978) 51–58. ⇒142

[17] Y. M. Shafransky and A. A. Doudkin, An optimization algorithm for the clearing of interbank payments, *European Journal of Operational Research*, 171, 3 (2006) 743–749. ⇒143

[18] V. Srinivasan and Y. H. Kim, Payments netting in international cash management: a network optimization approach, *Journal of International Business Studies*, 17, 2 (1986) 1–20. ⇒142, 143

[19] G. Syswerda, Schedule optimization using genetic algorithms, *Handbook of Genetic Algorithms*, 332–349, Van Nostrand Reingold, 1991. ⇒148

[20] A. J. Umbarkar and P. D. Sheth, Crossover operators in genetic algorithms: a review, *ICTACT journal on soft computing*, 6, 1, 2015 ⇒148

[21] T. Verhoeff, Settling multiple debts efficiently: an invitation to computing science, *Informatics in Education*, 3, 1 (2004), 105–126. ⇒143, 145, 146

[22] D. Whitley, T. Starkwater and D. Fuquay, Scheduling problems and traveling salesmen: The genetic edge recombination operator, *Proceedings of the Third International Conference on Genetic Algorithms*, 133–140, Morgan Kaufmann Publishers, 1989. ⇒147, 148

[23] J. Wróblewski, Theoretical foundations of order-based genetic algorithms, *Fundamenta Informaticae* **28,** 3-4 (1996) 423–430. ⇒148

Received: September 26, 2019 • Revised: November 26, 2019

Acta Univ. Sapientiae, Informatica **11,** 2 (2019) 159–173

DOI: 10.2478/ausi-2019-0011

On J-colorability of certain derived graph classes

Federico FORNASIERO
Department of Mathemathics
Universidade Federal de Pernambuco
Recife, Pernambuco, BRAZIL
email: federico@dmat.ufpe.br

Sudev NADUVATH
Department of Mathematics
CHRIST (Deemed to be University)
Bangalore-560029, INDIA.
email: sudev.nk@christuniversity.in

Abstract. A vertex v of a given graph G is said to be in a rainbow neighbourhood of G, with respect to a proper coloring $\mathcal{C}$ of G, if the closed neighbourhood $N[v]$ of the vertex v consists of at least one vertex from every color class of G with respect to $\mathcal{C}$. A maximal proper coloring of a graph G is a J-coloring of G such that every vertex of G belongs to a rainbow neighbourhood of G. In this paper, we study certain parameters related to J-coloring of certain Mycielski-type graphs.

1 Introduction

For general notations and concepts in graphs and digraphs we refer to [1, 3, 15]. For further definitions in the theory of graph coloring, see [2, 8, 4]. Unless specified otherwise, all graphs mentioned in this paper are simple, connected and undirected graphs.

Computing Classification System 1998: G.2.2
Mathematics Subject Classification 2010: 05C15, 05C38, 05C75.
Key words and phrases: Mycielski graphs, graph coloring, rainbow neighbourhoods, J-coloring of graphs.

1.1 Mycielskian of graphs

Let G be a triangle-free graph with the vertex set $V(G) = \{v_1, \dots, v_n\}$. The *Mycielski graph* or the *Mycielskian* of a graph G, denoted by $\mu(G)$, is the graph with vertex set $V(\mu(G)) = \{v_1, v_2, \dots, v_n, u_1, u_2, \dots, u_n, w\}$ such that $v_iv_j \in E(\mu(G)) \iff v_iv_j \in E(G)$, $v_iu_j \in E(\mu(G)) \iff v_iv_j \in E(G)$ and $u_iw \in E(\mu(G))$ for all $i = 1, \dots, n$ (see [9]).

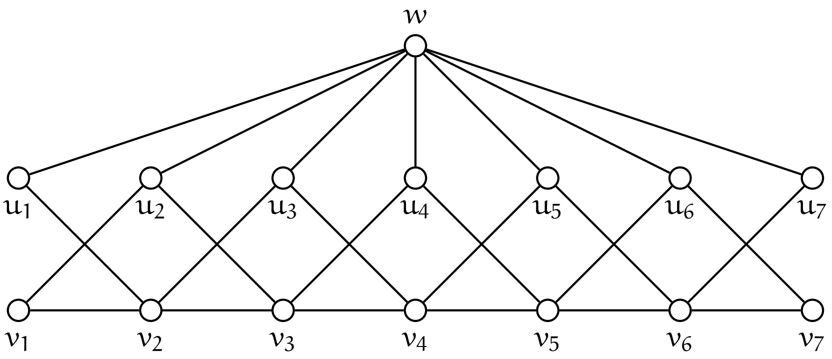


Figure 1: The Mycielski graph $\mu(P_7)$

In the above mentioned conditions of Mycielski graphs, we call the two vertices v_i, u_i *twin vertices* and the vertex w is called the *root vertex* of the Mycielskian $\mu(G)$.

By a *Mycielski type graph*, we mean a graph that can be constructed from the Mycielski graphs or the graphs generated from a given graphs using some or similar rules of constructing their Mycielski graphs.

1.2 Rainbow neighbourhoods in graphs

A *graph coloring* is an assignment of colors to its elements. If colors are assigned to the vertices of a graph G, then it is called a *vertex coloring* of G. A vertex coloring is said to be a *proper coloring* if no two adjacent vertices have the same color, with respect to the coloring concerned.

In this study, we follow a proper coloring protocol as follows: Assign color c_1 to the maximum possible number of vertices in G, then assign color c_2 is given to the maximum possible number of remaining uncolored vertices and the procedure is continued until all vertices of G are colored properly. Then, the closed neighbourhood $N[v]$ of a vertex $v \in V(G)$ which contains at least one colored vertex of each color with respect to the above-mentioned coloring, is called a *rainbow neighbourhood* in G. The number of rainbow neighbourhoods in a graph is said to be the *rainbow neighbourhood number* of the graph (see [5]).

Later rainbow neighbourhood number of different graph classes have been studied in detail and many interesting results have been added to the literature (see [5, 6, 7, 10, 11]). These studies motivated researchers to investigate further in this area and thus a new type of coloring called J-*coloring* has been introduced and studied.

1.3 J-coloring of graphs

The notion of J-coloring of a graph, has been defined for the first time in [12] as follows:

Definition 1 [12] A graph G is said to have a J-*coloring* $\mathcal{C}$ if it has the maximal number of colors such that every vertex v of G belongs to a rainbow neighbourhood of G. The number of colors in a J-coloring $\mathcal{C}$ of G is called the J-*coloring number* of G.

Definition 2 [12] A graph G is said to have a J^*-*coloring* $\mathcal{C}$ if it has the maximal number of colors such that every internal vertex v (a vertex with degree greater than 1) of G belongs to a rainbow neighbourhood of G. The number of colors in a J^*-coloring $\mathcal{C}$ of G is called the J^*-*coloring number* of G.

It can be noted that all graphs, in general, need not have a J-coloring. Hence, the studies on the graphs which admit J-coloring and their properties and structural characterisations attract much interests. Some studies in this direction can be seen in [6, 12, 13].

The initial purpose of this paper is to study the J-colorability of the Mycielskian and certain Mycielski type graphs of some fundamental graph classes.

2 J-colorability of Mycielski graphs

Note that the Mycielski graph $\mu(G)$ of a graph G has no pendant vertices and hence the J-coloring and the J^*-coloring of the Mycielski graphs $\mu(G)$, if they exist, are the same. We first try to repeat the original demonstration of Mycielskian graph. To do that, we have to fix a J-coloring on the graph G.

Theorem 3 *Let G be a graph with J-coloring $\mathcal{C} = \{c_1, c_2, c_3 \ldots, c_k\}$. Then, the graph $\mu(G)$ is not J-colorable (and so, it is not J^*-colorable).*

Proof. Note that $\mu(K_2)$ is isomorphic to C_5 and hence it is not $J-\texttt{colorable}$ (as it is proved in [12]). Hence, we can consider graphs with order greater than 2. Let us assume that we can have a new J-coloring $\mathcal{C} = \{c_1, c_2 \ldots, c_j\}$ on $\mu(G)$. Without loss of generality, we can assume now that the color of w is c_1'. Every vertex u_i have to be colored with one of the other color $c_2, \ldots, c_j$. Hence, there exists at least a vertex v_1 with color c_1. Let be v_2 the vertex connected to v_1 such as the twin vertex u_2 has the color c_2. Here, we have the following two possibilities:

(i) : If the color of v_2 is different from the previous ones, let us say c_3, we have that for the definition of rainbow neighbourhood even the twin vertex u_2 has to be connected with a vertex with the same color, and it has to be a vertex v_3 because no one of the vertices u_j are connected, and w has the color c_1. But if it is so, than for the construction of $\mu(G)$ even the vertex v_2 has to be connected with v_3, and so it is not a proper coloring because two vertices has the same colors (see Figure 2).

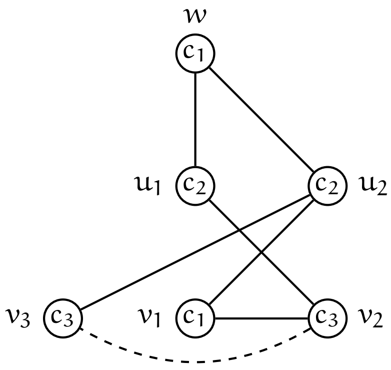


Figure 2: Case (i)

(ii) If the color of v_2 is c_2, the twin vertices u_1 of v_1 has to have a different color (let us say c_3), because it is linked to w that has the color c_1' and to v_2 that has the color c_2. But, in this case the vertex v_1 has to be connected with another vertex which has the color c_3.

So we have to differentiate two different possibilities:

(ii)(a) If the vertex v_1 is connected to a vertex v_3 who has the color c_3 for the construction of $\mu(G)$ even the twin vertices u_1 is connected to v_3 and so it is not a proper coloring because two connected vertices have the same color (see Figure 3).

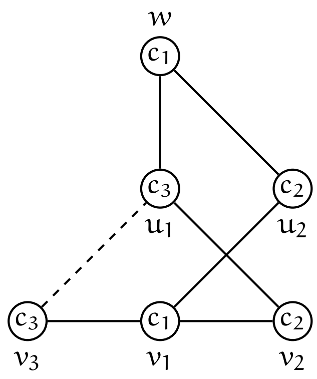


Figure 3: Case (ii)(a)

(ii)(b) If the vertex v_1 is connected to a vertex u_3 which has the color c_3, first we can note that the twin vertex v_3 cannot have a new color, because it would lead to a contradiction over u_3 similar to the case (i).

Note that v_3 cannot have the color c_1 (because for construction it is connected with v_1) nor the color c_3 (because always for construction it is connected to the twin vertex u_1) so it has to be colored with the color c_2. But if it is so, u_3 needs to be connected with a vertex v_i whose color is c_2, but it cannot be the twin vertex v_3 for the construction, nor the vertex v_2 because it will lead to have the triangle $v_1v_2u_3v_1$. If the graph G has only 3 nodes we reach a contradiction yet, if it is not let us call v_4 the vertex with color c_2 connected to u_3. For the construction of $\mu(G)$ it has to be connected to the vertex v_3 and it finally leads to a contradiction because the two vertices would have the same color (see Figure 4).

Hence, the Mycielskian graph of any graph G is not J-colorable, irrespective of whether the G is J-colorable or not. □

3 Some new constructions

Since Mycielskian of any graph does not admit a J-coloring, our immediate aim is to construct some simple connected graphs from certain given graphs

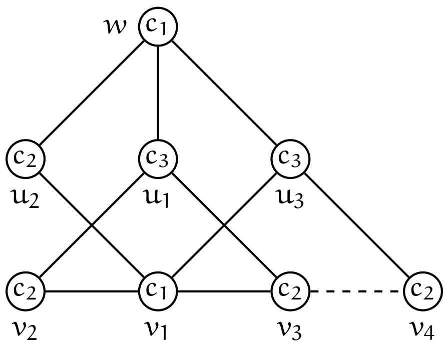


Figure 4: Case (ii)(b)

such that new graphs also admit an extended J-coloring. In this section, we discuss the J-colorability of certain newly constructed Mycielski type graphs of a given graphs.

The first one among such graphs is the *crib graph*, denoted by $c(G)$, of a graph G, which is defined in [13] as follows:

Definition 4 [13]The *crib graph*, denoted by $c(G)$, of a graph G is the graph whose vertex set is $V(\mu(G)) = \{v_1, v_2, \ldots, v_n, u_1, u_2, \ldots, u_n, w\}$ such that $v_iv_j \in E(\mu(G)) \iff v_iv_j \in E(G)$, $v_iu_j \in E(\mu(G)) \iff v_iv_j \in E(G)$ and $v_iw, u_iw \in E(\mu(G))$ for all $i = 1, \ldots, n$.

Figure 5 depicts the crib graph of P_6.

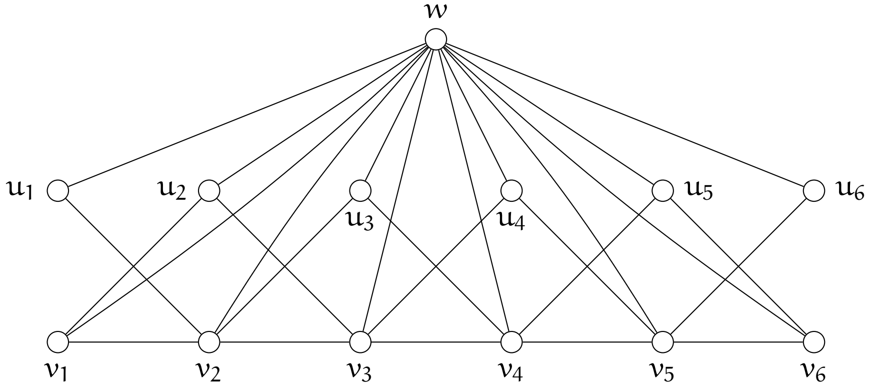


Figure 5: Crib graph of P_6

The following theorem discusses the admissibility of an extended J-coloring by the crib graph of a J-colorable graph G.

Theorem 5 *The crib graph* $c(G)$ *of a* J*-colorable graph* G *is also* J*-colorable. Also,* $\mathcal{J}(c(G)) = \mathcal{J}(G) + 1$.

Proof. Assume that the graph G under consideration admits a J-coloring, say $\mathcal{C} = \{c_1, c_2, \dots, c_k\}$, where $k = \chi(G)$, the chromatic number of G. While coloring the vertices of $c(G)$, we notice the following points:

(i) Since, the twin vertices u_i and v_i in $c(G)$ are adjacent to each other, both of them can have the same color.

(ii) Since $N(u_i) = N(v_i)$ for all $1 \leq i \leq n$, it follows that $N[u_i]$ is also a rainbow neighbourhood in $c(G)$. Therefore, the subgraph of $c(G)$ induced by the vertex set $\{v_1, v_2, \dots, v_n, u_1, u_2, \dots, u_n\}$ admit the same J-coloring $\mathcal{C}$.

(iii) Since the root vertex w is adjacent to other vertices in $c(G)$, it cannot have any color from $\mathcal{C}$. Therefore, we need a new color, say c_{k+1} to color the vertex w.

(iv) Since the root vertex w is adjacent to other vertices in $c(G)$, it belongs to a rainbow neighbourhood in $c(G)$ and will not influence the belongingness of other vertices to some rainbow neighbourhoods in $c(G)$.

In view of the conditions mentioned above, notice that $\mathcal{C} \cup \{c_{k+1}\}$ is a J-coloring of $c(G)$ and $\mathcal{J}(c(G)) = k + 1 = \mathcal{J}(G) + 1$. This completes the proof.
□

Another similar graph that catches attention in this context is the shadow graph of a graph G. The *shadow graph* of a graph G, denoted by $s(G)$, is the graph G is the graph obtained from its Mycielski graph $\mu(G)$ by removing the root vertex.
The following theorem discusses the admissibility of a J-coloring by the shadow graph $s(G)$ of a J-colorable graph G.

Theorem 6 *The shadow graph* $s(G)$ *of a* J*-colorable graph* G *is also* J*-colorable. Moreover,* $\mathcal{J}(s(G)) = \mathcal{J}(G)$.

Proof. The proof is immediate from the proof of Theorem 5. □

Next, we construct a new graph $F(G)$ from a triangle-free, simple and connected graph G such that $F(G)$ has J-chromatic number $k + 1$ when G has J-chromatic number k. The construction is described below.

Definition 7 Let G be a triangle-fee graph, with $V(G) = \{v_1, \dots, v_n\}$. We define the *Federico graph* $F(G)$ of G as the graph such that $V(F(G)) = \{v_1, v_2 \dots, v_n, u_1, u_2 \dots$ $u_n, w_1, w_2 \dots, w_n\}$ and with edges that follows the rules:

(i) $v_iv_j \in E(F(G)) \iff v_iv_j \in E(G)$

(ii) $w_iw_j \in E(F(G)) \iff v_iv_j \in E(G)$

(iii) $u_iw_j \in E(F(G)) \iff v_iv_j \in E(G)$

(iv) for all $i = 1, \dots, n$, $v_iu_i \in V(F(G))$

The following figure illustrates the Federico graph of the graph P_5.

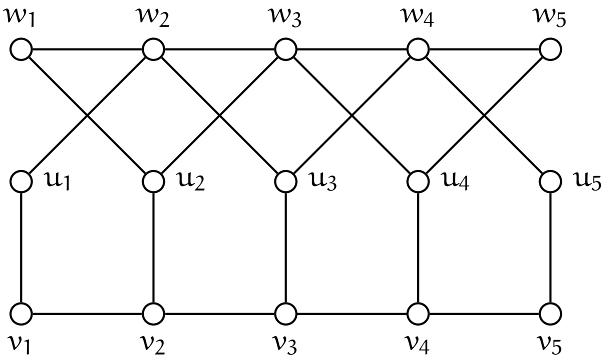


Figure 6: The Federico Graph $F(P_5)$

First we can note that the graph $F(G)$ has no pendant vertices and so the J-coloring od $F(G)$, if exists, coincides to the J^*-coloring. This fact is straight forward.

Theorem 8 *Let G be a J-colorable, triangle-free graph of order n with J-coloring number k. Then the graph $F(G)$ is triangle-free and with higher J-coloring number. If $\mathcal{J}(G) = k$, then $\mathcal{J}(F(G)) = k + 1$.*

Proof. First of all we can see that no pair of vertices u_i is connected, therefore no triangle can involve a pair of these vertices. Also, no vertex w_i is connected to a vertex v_j.

Remembering that G is triangle-free, it is not possible that three vertices v_i are connected in $F(G)$ too. Similarly for the vertices w_i that form between them a copy of the graph G. Hence, we have only two possibilities left:

(i) if v_i is connected to v_j we have that u_i is connected to v_i but not to v_j, by construction, so no triangle of this type is involved.

(ii) if w_i is connected to w_j we have that u_i is connected to w_j but not to w_i, so it is proved that $\mathsf{F}(\mathsf{G})$ is triangle-free.

To construct a proper J-coloring on $\mathsf{F}(\mathsf{G})$, let consider a proper J-coloring $\varphi : V(G) \to \{c_1, \dots, c_k\}$ and let us construct $\varphi^* : V(F(G)) \to \{c_1, \dots, c_k, c_{k+1}\}$ by setting:

(i) $\varphi^*(v_i) = f^*(w_i) = f(v_i)$ for all $i = 1, \dots, n$

(ii) $\varphi^*(u_i) = c_{k+1}$ for all $i = 1, \dots, n$

First, we have to prove that it is a proper J-coloring of $F(G)$. We note that every vertex v_i has a rainbow neighbourhood in G, and hence it has in $\mu(G)$ with one more color (the color of u_i). Every w_i has the same rainbow of the twin v_i because it is connected with the same vertices connected to v_i, and it is connected at least to one of the vertex u_j. Finally, every u_i has a $k+1$ rainbow neighbourhood of because it is connected with every v_i and with every w_j that are the connections of v_i in the original graph, so by the definition of φ^*, every u_i has the same rainbow neighbourhood of v_i. Hence, this coloring define a proper J-coloring of G, it remains to prove that this coloring is maximal.

Hence, let us assume that there exists a proper J-coloring of $F(G)$ such that $\mathcal{J}(F(G)) = 2$. In this case, we can assume that not every u_i has the same color because if not every u_i is connected only to v_i, and every v_i has a rainbow neighbourhood of order $k+2$ and can't have the same color of the vertices u_i. But it would mean that the graph G was $(k+1) - J$—coloring.

Hence, let us start considering the vertex u_i. If we prove that independent from the choice of the color of u_i, it is necessary that every u_i has the same color, for what we have just proved, it follows that the coloring is maximal.

From the above choice of the coloring assignment φ^*, we note that $F(G)$ requires at least one more color in its proper coloring than the corresponding proper coloring of the graph G. Now, note that the upper bound for the J-chromatic number of a graph G is $\delta(G)+1$ (see [12]). Since $\delta(F(G)) = \delta(G)+1$, any J-coloring of $F(G)$ can have at most one more color than the J-coloring of G. From these two conditions, we can conclude that the coloring φ^* defined above is a maximal coloring of $F(G)$ such that every vertex of $F(G)$ belongs to some rainbow neighbourhood of $F(G)$. Then, we have $\mathcal{J}(F(G)) = \mathcal{J}(G) + 1$, completing the proof. □

Hence, we have found an interesting construction to have new triangle free graphs with higher J-coloring number. In the following theorem, we study what happens to the chromatic number of a Federico graph.

Theorem 9 *Let* G *be a graph and* $F(G)$ *its Federico graph. Then,* $\chi(G) = \chi(F(G))$

Proof. Let $f : V(G) = \{v_1, \ldots, v_n\} \to c_1, \ldots, c_k$ be a coloring of the vertices of G. Let us consider the coloring $g : V(F(G)) = \{v_1, \ldots, v_n, u_1, \ldots, u_n, w_1, \ldots, w_n\} \to \{c_1, \ldots, c_k\}$ defined by:

(i) $g(u_i) = g(w_i) = f(v_i)$ for all $i = 1, \ldots, n$.

(ii) if $f(v_i) = c_h$ then $g(v_i) = c_{h+1}$ for all $i = 1, \ldots, n$ and $h = 1, \ldots, k-1$

(iii) if $f(v_i) = c_k$ then $g(v_i) = c_1$ for all $i = 1, \ldots, n$.

To prove that it is a proper coloring, first we can note that it is a proper coloring on the vertices v_i because f was a proper coloring of G and we have only permutated the colors, and also it is a proper coloring on the vertices w_i because it is a copy of the graph and we have colored in the same way. So it only left to see that we cannot have the same color with connections with a vertex u_i.

But, v_i is only connected to u_i and they have different colors because $g(u_i) = f(v_i)$ but $c_{h+1} = g(v_i) \neq f(v_i) = c_h$ for the definition of g. Also, because each w_i is connected to every u_j such that $v_j \in V(G)$ was connected to $v_i \in V(G)$ and none of which has the same color $g(u_i) = f(u_i)$ no conflicts arise here.

Hence, we have constructed a proper coloring of $F(G)$ with the same number of colors of G, as claimed. □

Now we want to study another important coloring property of the Modified Mycielski graph, the *circular chromatic number*. It was first studied in [14] with the name of star chromatic number, and later in [16] provided a comprehensive survey.

Let G be a graph. For two positive numbers k, d with $k \geq 2d$, we define a (k, d)-coloring as the function $f : V(G) \to \{0, 1, \ldots, k-1\}$ such that if two vertices u, v are adjacent, then $|f(u) - f(v)|_k \leq d$ where $|a - b|_k = \min\{|a - b|, k - |a - b|\}$. Then, the *circular chromatic number* of G is defined as

$$\chi_c(G) := \inf \left\{ \frac{k}{d} \mid G \text{ has a } (k, d) - \text{coloring} \right\}$$

In [16] it is shown that if the graph G has at least one edge, then the infimum can be replaced with the minimum and we have $\chi(G) - 1 \leq \chi_c(G) \leq \chi(G)$.

The circular chromatic number is hard to compute in Mycielski graphs and there's not yet a general formula that compute $\chi_c(\mu(G))$ knowing the circular chromatic number of G. But, in the case of Federico graph, we have

Theorem 10 *Let* G *be a graph with* $\chi_c(G)$*. Then,* $\chi_c(F(G)) = \chi_c(G)$*.*

Proof. Let G be a graph with a (k,d) coloring f over $V(G) = \{v_1, v_2 \dots, v_n\}$. Then, we construct the coloring f^* over $F(G)$ as follows:

(i) $f^*(v_i) = f(v_i)$ for all $i = 1, \dots, n$

(ii) $f^*(u_i) = f^*(w_i) = f(v_i) - d \bmod k$ for all $i = 1, \dots, n$

To see that it's a proper (k,d) coloring of $F(G)$ we first note that between it's a proper coloring over the vertices v_i's (because it was on G) and over the vertices w_i's (because in the construction we have simply added the distance modulo k, and their connections are the same than the connections over the vertices v_i). A vertex v_i is adjacent only to the vertex u_i and so by construction it has exactly distance d.

The vertex u_i is connected to every vertex w_j such that v_iv_j is an edge in G. But for construction the vertex u_i has color $f(v_i) - d \bmod k$ and the vertices w_j have color $f(v_j) - d \bmod k$, so the connection maintain the same distances over the edges $v_iv_j \in E(G)$. Therefore, it is a proper (k,d)—coloring of $F(G)$ and if $\frac{k}{d}$ is minimal in G and hence it is minimal in $F(G)$. □

4 J-paucity number of graphs

In view of our results on the absence of J-coloring for Mycielski graphs and our new constructions from the Mycielski graphs which admit J-colorings, we define a new graph parameter with respect to J-coloring as follows:

Definition 11 Let G be a graph which does not admit a J-coloring. Then, the J-*paucity number* of G, denoted by $\rho(G)$, is defined as the minimum number of edges to be added to G so that the reduced graph becomes J-colorable with respect to a $(\delta(G)+1)$-coloring of G.

In the following theorem, we determine the J-paucity number of paths.

Theorem 12 $\rho(\mu(P_n)) = n$*.*

Proof. Note that for $\delta(\mu(P_n)) = 2$ and hence we have to find the minimum number of edges to be added to $\mu(P_n)$ so that the reduced graph becomes J-colorable using 3 colors. For this, first assign colors c_1 and c_2 alternatively to the vertices $v_1, v_2, \dots, v_n$. Now color the vertices u_i such that u_i and its twin vertex v_i have the same color. Since the vertex w is adjacent to all u_i's, it can be seen that it must have a different color, say c_3 (see Figure 7).

We notice the following points in this context:

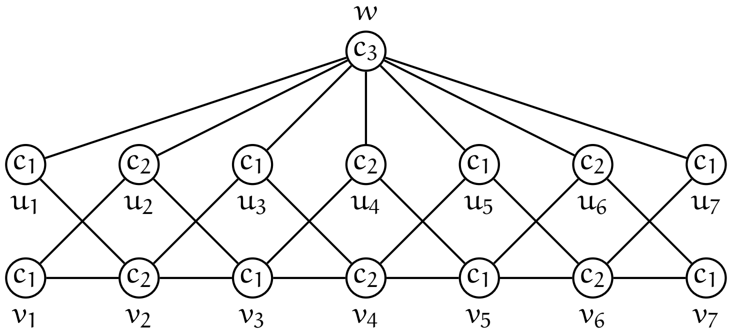


Figure 7: A 3-coloring of $\mu(P_7)$.

(i) No vertex v_i in $V(\mu(P_n))$ belongs to a rainbow neighbourhood of $\mu(P_n)$, as none of them is adjacent to a vertex having color c_3;

(ii) Every vertex u_i with color c_2 is adjacent to at least one vertex v_j with color c_2 and the vertex w with color c_3, thus belonging to some rainbow neighbourhood in $\mu(P_n)$.

(iii) Every vertex u_j with color c_2 is adjacent to at least one vertex v_k with color c_1 the vertex w with color c_3, thus belonging to some rainbow neighbourhood in $\mu(P_n)$.

(iv) The vertex w, being adjacent to all vertices u_i, belongs to some rainbow neighbourhoods in $\mu(P_n)$.

Therefore, from the above arguments, what we need is to draw edges from the vertices v_i to the vertex w so that they also are in some rainbow neighbourhoods of G. Therefore, $\rho(\mu(P_n)) = n$. □

Theorem 13 $\rho(\mu(C_n)) = n + 2r$, *where* $r \in \mathbb{N}$ *is given by* $n \equiv r (\text{mod}\, 3)$.

Proof. Since $\delta(\mu(C_n)) = 3$, the maximum number of colors in its J-coloring is 4. Hence, we have to find the minimum number of edges to be added to $\mu(P_n)$ so that the reduced graph becomes J-colorable using 4 colors. Here we have to consider the following cases:

Case-1: Let $n \equiv 0 (\text{mod}\, 3)$. Then, we can assign colors c_1, c_2 and c_3 alternatively to the vertices $v_1, v_2, \ldots, v_n$. As mentioned in the previous result, we can color the vertices u_i such that u_i and its twin vertex v_i have the same color. Since the vertex w is adjacent to all u_i's, it must have a different color, say c_4 (see Figure 8). In this case, all vertices u_i and the vertex w will belong to some rainbow neighbourhoods of $\mu(C_n)$, but no vertex v_i has an adjacent vertex having color c_4. So, we need to draw edges from all v_i; $1 \leq i \leq n$ to the vertex w in order to include them in some rainbow neighbourhoods of $\mu(C_n)$.

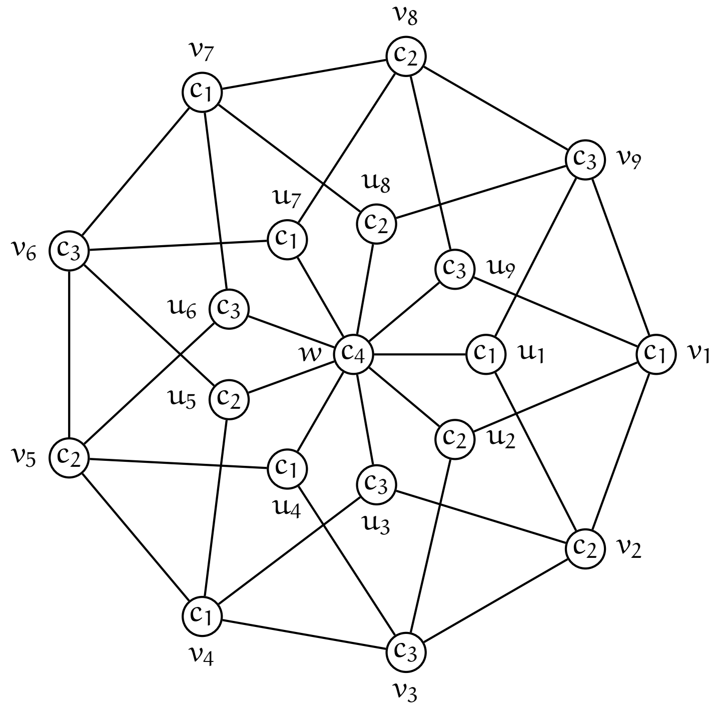


Figure 8: A minimal proper coloring of $\mu(C_9)$

Case-2: Let $n \equiv 1 (\operatorname{mod} 3)$. Then, we can assign colors c_1, c_2 and c_3 alternatively to the vertices $v_1, v_2, \ldots, v_{n-1}$. The vertex v_n can be colored only by c_2, as it is adjacent to v_1 with color c_1 and to v_{n-1} with color c_3. Here, we notice that the vertex v_1 is not adjacent to any vertex having color c_3. Here, we need to draw an edge between v_1 and one of the vertices having color c_3.

If we label the vertices u_i in such a way that the twin vertices have the same color, then as in the case of v_1, the vertex u_1 is not adjacent to any vertex of color c_3. Hence, we need to draw an edge from u_1 to any one of the vertices having color c_3.

Since w is adjacent to all u_i, w must have the fourth color c_4. Since every vertex u_i is adjacent to w, all these vertices,(except u_1) belong to some rainbow neighbourhood of $\mu(C_n)$. (Also, note that when we draw an edge from u_1 to a vertex having color c_3, it will also belong to some rainbow neighbourhood).

Since no vertex v_i is adjacent to a vertex having color c_4, each of them is to be connected to the vertex w by a new edge. Therefore, in this case $\rho(\mu(C_n)) = n + 2$.

Case-3: Let $n \equiv 2 (\mathrm{mod}\, 3)$. Then, we can assign colors c_1, c_2 and c_3 alternatively to the vertices $v_1, v_2, \ldots, v_{n-2}$. Then, the vertex v_{n-1} gets the color c_1, the vertex v_1 can have the color color c_2 Note that the vertex v_1 and v_n are not adjacent to any vertex having color c_3. Here, we need to draw one edge each from v_1 and v_2 to some vertices having color c_3.

If we label the vertices u_i in such a way that the twin vertices have the same color, then as in the case of v_1 and v_2, the vertices u_1 and u_n will not be adjacent to any vertex of color c_3. Hence, we need to draw one edge each from u_1 and u_3 to some of the vertices having color c_3.

The vertex w gets the color c_4 and as mentioned in the above cases, we need to draw edges from all vertices v_i to w so that all vertices in $\mu(C_n)$ belong to some rainbow neighbourhoods in $\mu(C_n)$. Therefore, in this case $\rho(\mu(C_n)) = n + 4$. □

5 Conclusion

In this paper, we have proved that the Mycielskian of any graph G will not have a J-coloring, irrespective of whether G has a J-coloring or not. We have also checked the existence of J-coloring for certain new Mycielski type graphs constructed from certain graphs. There is a wide scope for further studies in this area by exploring for new and related graph constructions.

We have also investigated the possibility of defining J-colorings for given graphs by adding new edges between their non-adjacent vertices. Furthermore, we have determined the minimum number of such edges to be introduced for the Mycielskian of paths and cycles. The studies in this area for more graph classes and more derived graphs are also promising.

Acknowledgements

Authors would like to acknowledge the creative and critical commends of the referee which improved the content and the presentation of the papersignificantly.

References

[1] J.A. Bondy, U.S.R. Murty, *Graph theory*, Springer, Berlin, 2008. ⇒159

[2] G. Chartrand, P. Zhang, *Chromatic graph theory*, CRC Press, Bocca Raton, 2009. ⇒159

[3] F. Harary, *Graph theory*, Narosa Publishing House, New Delhi, 2001. ⇒159

[4] T. R. Jensen, B. Toft, *Graph coloring problems*, John Wiley & Sons, 1995. ⇒ 159

[5] J. Kok, N. K. Sudev, U. Mary, On chromatic Zagreb indices of certain graphs, *Discrete Math. Algorithm. Appl.*, **9,** 1 (2017) 1750014:1–14, DOI: 10.1142/S17938309175001 48. ⇒161

[6] J. Kok, N. K. Sudev, J-coloring of graph operations, *Acta Univ. Sapientiae Inform.*, **11,** 1 (2017) 95–108. ⇒161

[7] J. Kok, N. K. Sudev, M. K. Jamil, Rainbow neighbourhood number of graphs, *Proyecciones J. Math.*, **39,** 3 (2019) 469–485. ⇒161

[8] M. Kubale, *Graph colorings*, American Mathe. Soc., 2004. ⇒159

[9] W. Lin, J. Wu, P. C. B. Lam, G. Gu, Several parameters of generalized Mycielskians, *Discrete Appl. Math.*, **154,** 8 (2006) 1173–1182, DOI:10.1016/j.dam.2005.11.001. ⇒160

[10] N. K. Sudev, C. Susanth, S. J. Kalayathankal, J. Kok, A note on the rainbow neighbourhood number of graphs, *Nat. Acd. Sci. Lett.*, **43,** 2 (2019) 135–138. ⇒ 161

[11] N. K. Sudev, C. Susanth, S. J. Kalayathankal, J. Kok, Some new results on the rainbow neighbourhood number of graphs, *Nat. Acd. Sci. Lett.*, **43,** 2 (2019) 249–252. ⇒161

[12] N.K. Sudev, On certain J-coloring parameters of graphs, *Nat. Acad. Sci. Lett.*, to appear. ⇒161, 162, 167

[13] C. Susanth, S.J. Kalayathankal, N.K. Sudev, Rainbow neighbourhood number of certain Mycielski type graphs, *Int. J. Appl. Math.*, **31,** 6 (2018) 797–803. ⇒ 161, 164

[14] A. Vince, Star chromatic number, *J. Graph Theory* **12** (1988), 551–559. ⇒168

[15] D. B. West, *Introduction to graph theory*, Pearson Education Inc., 2001. ⇒159

[16] X. Zhu, The circular chromatic number: A survey, *Disc. Math.*, **229,** (2001) 371–410. ⇒168

Received: October 22, 2019 • Revised: November 30, 2019

Acta Univ. Sapientiae, Informatica **11,** 2 (2019) 174–183

DOI: 10.2478/ausi-2019-0012

Complexity of domination in triangulated plane graphs

Dömötör PÁLVÖLGYI
MTA-ELTE Lendület Combinatorial Geometry
Research Group, Institute of Mathematics, Eötvös
Loránd University (ELTE), Budapest, Hungary
email: dom@cs.elte.hu

Abstract. We prove that for a triangulated plane graph it is NP-complete to determine its domination number and its power domination number.

1 Introduction

Given a graph $G = (V, E)$, for a subset of the vertices $S \subset V$, denote by $\Gamma(S)$ the closed neighborhood of S, i.e.,

$$\Gamma(S) = S \cup \{v \in V \mid \exists\, s \in S \textit{ such that } (v, s) \in E\}.$$

S is called a *dominating set* if $V = \Gamma(S)$, i.e., every vertex from $V \setminus S$ has a neighbor in S. The size of the smallest dominating set is called the *domination number* of G and is denoted by $\gamma(G)$. A simple graph embedded in the plane without crossing edges is called a *triangulated plane graph* if each of its faces (including the other face) is *triangular*, i.e., its boundary consists of three edges. We emphasize that in this paper we only consider undirected *simple* graphs, i.e., multiple edges are not allowed. Garey and Johnson [5] have proved that it is NP-hard to determine $\gamma(G)$, already for planar graphs. We extend this result to triangulated planar graphs.

Computing Classification System 1998: G.2.2
Mathematics Subject Classification 2010: 05C10
Key words and phrases: NP-completeness, planar graphs, domination

Theorem 1 *For a triangulated plane graph* G *and integer* n*, it is* NP*-complete to determine its domination number, that is, to decide whether* $\gamma(G) \leq n$*.*

Our method also works for the related parameter called *power domination number*. This problem originates from monitoring electrical networks with so-called Phasor Measurement Units; it was first formulated for graphs by Haynes et al. [7], but we use the (somewhat different) definition given by Brueni and Heath [3]. Given a graph $G = (V, E)$, a set of vertices S, let S_1 be the subset of vertices from S that have exactly one neighbor outside S, i.e.,

$$S_1 = \{s \in S \mid \exists!\ v \in V \setminus S \textit{ such that } (s, v) \in E\}.$$

The vertices of S_1 can *propagate* to their neighbors, so we define

$$\Gamma_1(S) = S \cup \Gamma(S_1).$$

The *power domination process* starts from any set of vertices S, in the first steps applies the Γ operator, and then in each following step the Γ_1 operator, until Γ_1 stops increasing the size of the set (which happens after finitely many steps in a finite graph). The set of vertices obtained this way is denoted by

$$\Gamma_P(S) = \Gamma_1(\dots\Gamma_1(\Gamma(S))\dots).$$

If $V = \Gamma_P(S)$, then we say that S is a *power dominating set* and the size of the smallest such set is the *power domination number*, $\gamma_P(G)$, of the graph G. Brueni and Heath [3] have proved that it is NP-hard to determine $\gamma_P(G)$, already for planar graphs. We extend this result to triangulated planar graphs.

Theorem 2 *For a triangulated plane graph* G *and integer* n*, it is* NP*-complete to determine its power domination number, that is, to decide whether* $\gamma_P(G) \leq n$*.*

In fact, our construction will be such that either there is an S with $|S| = n$ such that already $V = \Gamma_1(\Gamma(S))$, or $\gamma_P(G) > n$.

For more related literature and background, see the recent works [1, 4].

2 Technical claims

Our reductions are from the Planar Monotone 3-sat problem, which was defined and shown to be NP-complete in [2]. In this problem the goal is to

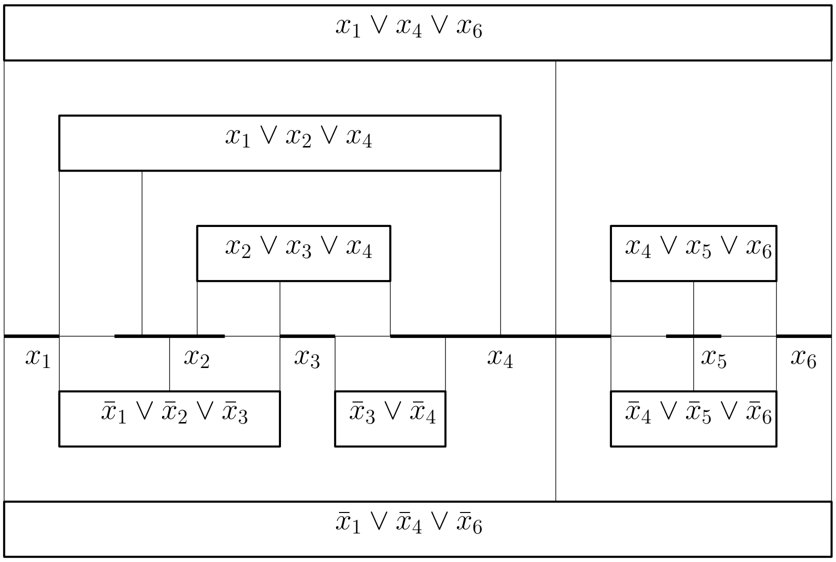

$x_1 \vee x_4 \vee x_6$
$x_1 \vee x_2 \vee x_4$
$x_2 \vee x_3 \vee x_4$
$x_4 \vee x_5 \vee x_6$
x_1
x_2
x_3
x_4
x_5
x_6
$\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3$
$\bar{x}_3 \vee \bar{x}_4$
$\bar{x}_4 \vee \bar{x}_5 \vee \bar{x}_6$
$\bar{x}_1 \vee \bar{x}_4 \vee \bar{x}_6$

Figure 1: Example of a Planar Monotone 3-sat input. A satisfying assignment: only x_4 is true.

decide the satisfiability of a conjunctive normal form (CNF), where each clause contains at most 3 literals, all of which are either negated, or all unnegated, along with a planar embedding of the incidence structure in the following way. (See Figure 1.)

- Each variable corresponds to an interval in the horizontal line $y = 0$; these intervals are pairwise disjoint.
- Each clause corresponds to an axis-parallel rectangle; these rectangles are pairwise disjoint.
- If a clause contains only negated (resp. unnegated) variables, then its rectangle is entirely contained in the $y < 0$ (resp. $y > 0$) halfplane.
- Every rectangle is connected to (the intervals corresponding to) the variables contained in (the clause corresponding to) it by a vertical segment, which does not pass through any other rectangles.

Note that clauses containing less than 3 literals are also allowed; we are not aware of whether the problem remains NP-complete or not if we require that every clause contains exactly 3 literals (this would slightly simplify our proof). Note that without requiring monotonicity (and any other special structure)

Planar Exact 3-sat is NP-complete [9], even if the planar incidence graph is vertex 3-connected [8]. In our case, however, it seems more likely that the problem always becomes solvable. This would also follow from a conjecture of Goddard and Henning [6], according to which the vertices of any plane triangulation can be 2-colored such that each vertex is adjacent to a vertex of each color. (Here we do not go into details about why their conjecture would imply our claim; it involves a triangulation similar to the one that can be found in our main proof.)

We can, however, suppose that no clause contains exactly 1 literal, as in this case the formula could be easily simplified. Moreover, we can also suppose that if a clause contains exactly 2 literals, then there is no other clause that would contain the same two literals (with the same negations); e.g., $(x_i \vee x_j) \wedge (x_i \vee x_j \vee x_k)$ is equivalent to $(x_i \vee x_j)$. Because of this, and the properties of the embedding, we can suppose the following.

Observation 3 *For any two literals there are at most two clauses that contain both of them, and if two such clauses exist, both of them also contains a third literal.*

We will also use the following technical lemma about triangulating plane graphs.

Lemma 4 *Suppose that* $G = (V, E)$ *is a plane graph and* $Z \subset V$ *is a subset of its vertices such that*

(1) every vertex $z \in Z$ *has at least three neighbors,*

(2) for a vertex $z \in Z$ *and two of the edges adjacent to it,* (z, v) *and* (z, v')*, that follow each other in the rotation around* z *in the embedding of* G*, either* $(v, v') \notin E$ *or* (v, v', z) *forms a triangular face,*

(3) if $z, z' \in Z$ *are neighbors, then they have exactly two common neighbors,* $v, v' \in V$*, and* (z, z', v) *and* (z, z', v') *are two triangular faces of the embedding,*

(4) if two vertices $v, v' \in V \setminus Z$ *have two common neighbors from* Z*, then they have exactly two common neighbors from* Z*,* z *and* z'*, and* (v, z, v', z') *is a face of the embedding of* G*,*

then G *can be triangulated by adding only edges that are not adjacent to any vertex in* Z*.*

Proof. We need to show that if for a vertex $z \in Z$ two of the edges adjacent to it, (z, v) and (z, v'), follow each other in the embedding of G in the rotation around z, then either $(v, v') \in E$ and (v, v', z) forms a triangular face, or (v, v') can be added as such. This way the faces around each $z \in Z$ become triangulated and we can triangulate the rest of the graph arbitrarily.

We handle the following cases separately.

- If $(v, v') \in E$, then because of condition (2) (v, v', z) forms a triangular face.
- If v or v' is from Z, then because of condition (3) $(v, v') \in E$.
- If v and v' have no other common neighbor from Z, then connect them by an edge in the vicinity of the curves of the edges (v, z) and (z, v').
- If v and v' have another common neighbor from Z, then because of condition (4) they have exactly one, $z' \in Z$, and (v, z, v', z') is a face of the embedding of G, thus we can divide it by adding the edge (v, v').

By repeatedly applying the above, the only condition we could violate is condition (2) by adding the edge (v, v') such that (v, v', z) does not form a triangular face. But we can add (v, v') to G only in the last two cases, when v and v' have a common neighbor from Z, and in each case (v, v', z) forms a triangular face after adding (v, v'). This finishes the proof of Lemma 4. □

3 Proofs of Theorems 1 and 2

Proof. [of Theorem 1] The problem is trivially in NP, we only have to prove its hardness.

Given an input Ψ to the Planar Monotone 3-sat problem on n variables, we transform it into a plane triangulation G such that $\gamma(G) \leq n$ if and only if Ψ is satisfiable. (See Figure 2 for the basic graph G obtained from Ψ and Figure 3 for the plane triangulation.)

For each variable x_i, G will contain a K_4 (a complete graph on 4 vertices), whose vertices we denote by $v_i, \bar{v}_i, u_i, w_i$. The vertex w_i has no other neighbors, which already shows that $\gamma(G) \geq n$, as we must select a vertex from each K_4.

For each clause C_h we introduce a vertex, z_h, that is connected only to one vertex for each literal it contains; if $x_i \in C_h$, then we connect z_h to v_i, while if $\bar{x}_i \in C_h$, then we connect z_h to $\bar{v}_i$.

The graph obtained so-far is obviously planar, now we need the following bound on its domination number.

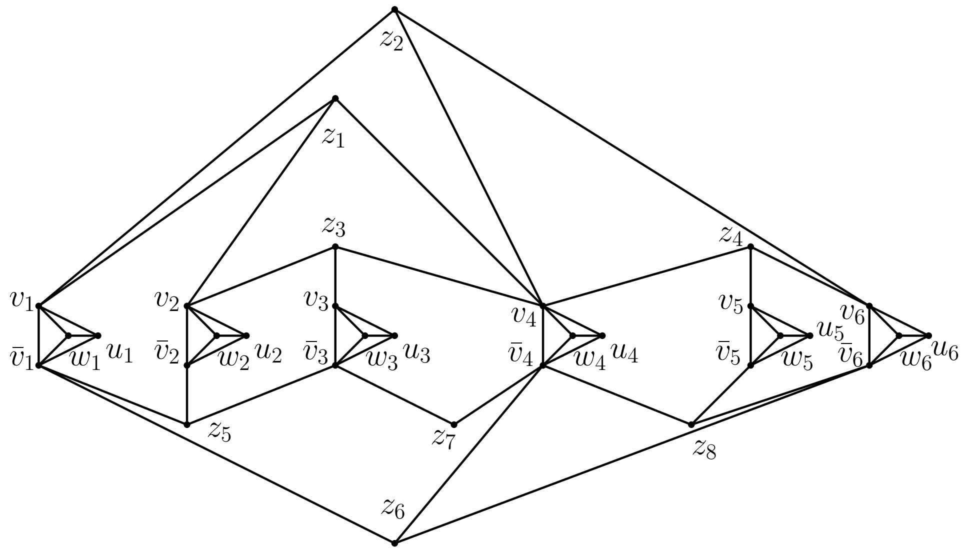


Figure 2: Example of graph G used for hardness of domination obtained from Planar Monotone 3-sat input. A dominating set of size 6: $\{v_4, \bar{v}_1, \bar{v}_2, \bar{v}_3, \bar{v}_5, \bar{v}_6\}$.

Claim 5 $\gamma(G) = n$ *if and only if* Ψ *is satisfiable.*

Proof. Suppose that Ψ is satisfiable and fix a satisfying assignment. If x_i is true, we can let $v_i \in S$, and if x_i is false, we can let $\bar{v}_i \in S$. This way we have picked a vertex from each K_4 corresponding to the variables and since the assignment satisfies Ψ, every vertex z_h corresponding to a clause is also dominated.

Suppose that $\gamma(G) = n$ and fix a dominating set S of size n. As w_i needs to be dominated for each i, $|S \cap \{v_i, \bar{v}_i, u_i, w_i\}| = 1$. If $v_i \in S$, we can let x_i be true, if $\bar{v}_i \in S$, we can let x_i be false, and otherwise we can choose its truth value arbitrarily. This way each clause is satisfied, as the corresponding vertex z_h had to be dominated. $\square$

This already establishes the hardness of the problem for plane graphs; to finish the proof of Theorem 1, we only need to show that we can triangulate G without introducing any new neighbors to the z_h vertices. If each clause of Ψ contains exactly three literals, then this follows by taking the (not necessarily straight-line) “natural embedding” of G obtained from the embedding of Ψ, and applying Lemma 4 with Z containing the z_h vertices that correspond to the clauses (it is straight-forward to check that the conditions of Lemma 4

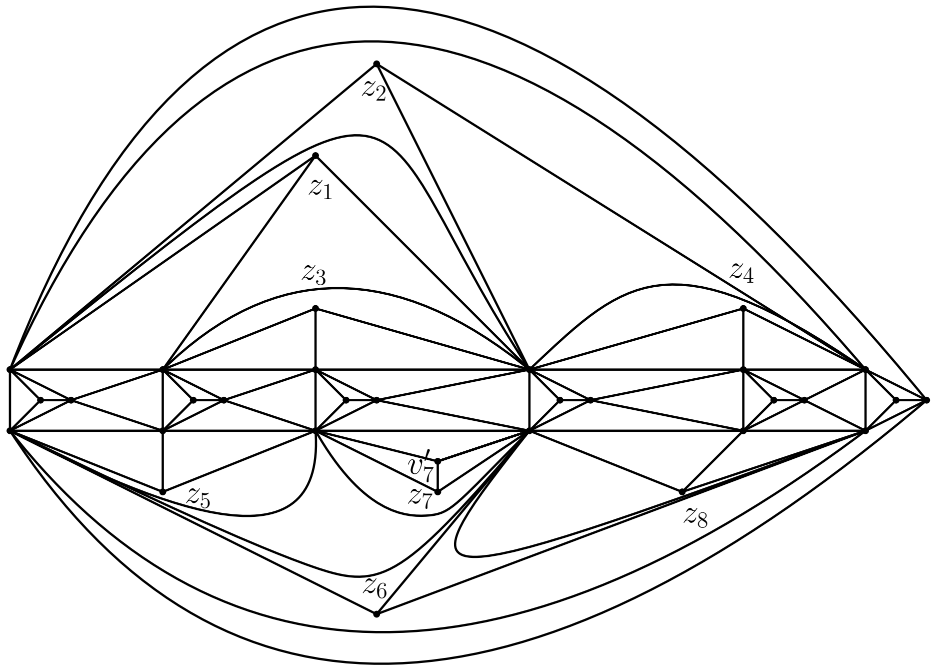


Figure 3: Triangulation of G (with vertex v_7' added to the only clause with two variables).

hold using Observation 3).

If Ψ also contains clauses with only two literals, we need to introduce extra vertices to G in the following way. For each clause with two literals, e.g., $C_h = (x_i \vee x_j)$, we add one extra vertex, v_h', that we connect to x_i, x_j and z_h. Note that this does not change the domination number of G, as v_h' is connected to exactly the same vertices as z_h, and they are also connected to each other. But now the conditions of Lemma 4 hold with Z containing the z_h vertices, thus we can obtain a triangulation, finishing the proof of Theorem 1. □

Proof. [of Theorem 2] As in the case of Theorem 1, the problem is trivially in NP, we only have to prove its hardness.

Given an input Ψ to the PLANAR MONOTONE 3-SAT problem on n variables, we transform it into a plane triangulation G such that $\gamma_P(G) \leq n$ if and only if Ψ is satisfiable. (See Figure 4.)

For each variable x_i, G will contain six vertices, $v_i, \bar{v}_i, u_i, v_i', \bar{v}_i', u_i'$, such that they all have edges between them except (v_i, v_i'), $(\bar{v}_i, \bar{v}_i')$ and (u_i, u_i'). The vertices $v_i', \bar{v}_i', u_i'$ have no other neighbors among the other vertices of

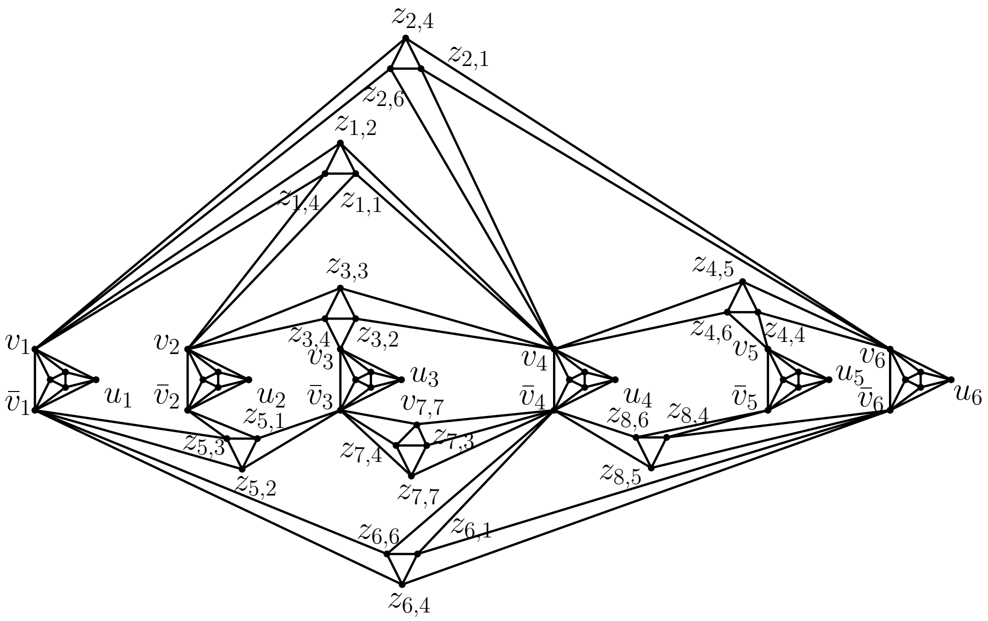


Figure 4: Example of graph G used for hardness of power domination obtained from PLANAR MONOTONE 3-SAT input. A power dominating set of size 6: $\{v_4, \bar{v}_1, \bar{v}_2, \bar{v}_3, \bar{v}_5, \bar{v}_6\}$. This graph can be triangulated similarly as on Figure 3.

the graph, thus their degrees are 4. This already shows that $\gamma(G) \geq n$, as we must select a vertex from each such sextuple[1], otherwise we could not propagate to $v'_i, \bar{v}'_i, u'_i$, as each of their neighbors is adjacent to at least two of them. If, however, we choose any of $v_i, \bar{v}_i, u_i$ to our initial set S, we have $\{v_i, \bar{v}_i, u_i, v'_i, \bar{v}'_i, u'_i\} \subset \Gamma_1(\Gamma(S)) \subset \Gamma_P(S)$.

For each clause with three literals, e.g., $C_h = (x_i \vee x_j \vee x_k)$, we introduce three degree 4 vertices, $z_{h,i}, z_{h,j}, z_{h,k}$, that are connected to each other and to two of the literals each; $z_{h,i}$ is connected to v_j and v_k, $z_{h,j}$ is connected to v_i and v_k, and $z_{h,k}$ is connected to v_i and v_j. (If C_h contained negated literals, than instead of the v_i, v_j, v_k we would use $\bar{v}_i, \bar{v}_j, \bar{v}_k$.) Notice that we must select

[1]The six titles won by Barcelona in 2009 (Copa del Rey, La Liga, UEFA Champions League, Supercopa de España, UEFA Super Cup and FIFA Club World Cup) have been described as a 'sextuple'. This achievement, however, took place over the course of two different Spanish seasons, including a treble in the 2008-09 season. Despite occurring in two seasons, the six titles are still counted as a 'sextuple' by many people, because the three added trophies (during the 2009-2010 season) were extra matches of the 2008-2009 treble and all six titles were won in the same calendar year.

at least one of $v_i, v_j, v_k, z_{h,i}, z_{h,j}, z_{h,k}$, otherwise we could not propagate to $z_{h,i}, z_{h,j}, z_{h,k}$, as each of their neighbors is adjacent to at least two of them. If, however, we choose any of v_i, v_j, v_k to our initial set S, we have $\{z_{h,i}, z_{h,j}, z_{h,k}\} \subset \Gamma_1(\Gamma(S)) \subset \Gamma_P(S)$.

For each clause with two literals, e.g., $C_h = (x_i \vee x_j)$, we introduce four degree 4 vertices, $z_{h,i}, z_{h,j}, z_{h,h}, v_{h,h}$, that are connected to each other and two additional vertices each: $z_{h,i}$ is connected to v_j and $v_{h,h}$, $z_{h,j}$ is connected to v_i and $v_{h,h}$, and $z_{h,h}$ and $v_{h,h}$ are connected to v_i and v_j. (If C_h contained negated literals, than instead of the v_i and v_j we would use $\bar{v}_i$ and $\bar{v}_j$.) Notice that we must select at least one of $v_i, v_j, z_{h,i}, z_{h,j}, z_{h,h}, v_{h,h}$, otherwise we could not propagate to $z_{h,i}, z_{h,j}, z_{h,k}$, as each of their neighbors is adjacent to at least two of them. If, however, we choose any of v_i or v_j to our initial set S, we have $\{z_{h,i}, z_{h,j}, z_{h,h}, v_{h,h}\} \subset \Gamma_1(\Gamma(S)) \subset \Gamma_P(S)$.

The graph obtained so-far is obviously planar, now we need the following bound on its domination number.

Claim 6 $\gamma_P(G) = n$ *if and only if* Ψ *is satisfiable.*

Proof. Suppose that Ψ is satisfiable and fix a satisfying assignment. If x_i is true, we can let $v_i \in S$, and if x_i is false, we can let $\bar{v}_i \in S$. This way we have picked a vertex from each sextuple corresponding to the variables and since the assignment satisfies Ψ, every vertex corresponding to a clause is power dominated by S.

Suppose that $\gamma(G) = n$ and fix a power dominating set S of size n. As we need to pick a vertex from each sextuple for each i, $|S \cap \{v_i, \bar{v}_i, u_i, v_i', \bar{v}_i', u_i'\}| = 1$. If $v_i \in S$, we can let x_i be true, if $\bar{v}_i \in S$, we can let x_i be false, and otherwise we can choose its truth value arbitrarily. This way each clause is satisfied, as the corresponding $z_{h,.}$ vertices had to be power dominated. □

This already establishes the hardness of the problem for plane graphs; to finish the proof of Theorem 2, we only need to show that we can triangulate G without introducing any new neighbors to the z_h vertices. This follows by taking the "natural embedding" of G obtained from the embedding of Ψ, and applying Lemma 4 with Z containing the $z_{h,.}$ vertices that correspond to the clauses (it is straight-forward to check that the conditions of Lemma 4 hold using Observation 3). □

Acknowledgment

I am thankful to the organizers of the Workshop on Graph and Hypergraph Domination and its participants where this research was started, especially to

Paul Dorbec for proposing the problem of the complexity of determining the power domination number in triangulated plane graphs and to Claire Pennarun for useful discussions, providing references and for calling my attention to the related problem about the domination number.

Research supported by the Marie Skłodowska-Curie action of the European Commission, under grant IF 660400, and by the Lendület program of the Hungarian Academy of Sciences (MTA), under grant number LP2017-19/2017.

References

[1] A. Aazami, Domination in graphs with bounded propagation: algorithms, formulations and hardness results, *J. Comb. Optim.*, **19,** 4 (2010) 429–456. ⇒175

[2] M. de Berg and A. Khosravi, Optimal binary space partitions for segments in the plane, *Int. J. Computational Geometry & Applications* **22** (2012) 187–206. ⇒175

[3] D. J. Brueni and L. S. Heath, The PMU placement problem, *SIAM Journal on Discrete Mathematics* **19,** 3 (2005) 744–761. ⇒175

[4] P. Dorbec, A. González, C. Pennarun, Power domination in maximal planar graphs, *manuscript*, arXiv:1706.10047 (2017). ⇒175

[5] M. R. Garey and D. S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*. W. H. Freeman and Co. San Francisco, California: W. H. Freeman and Co. pp. x+338. ISBN 0-7167-1045-5. ⇒174

[6] W. Goddard, M. A. Henning, Thoroughly Distributed Colorings, *manuscript*, arXiv:1609.09684 (2016). ⇒177

[7] T. W. Haynes, S. M. Hedetniemi, S. T. Hedetniemi, and M. A. Henning, Domination in graphs applied to electric power networks, *SIAM Journal on Discrete Mathematics* **15,** 4 (2002) 519–529. ⇒175

[8] J. Kratochvíl, A special planar satisfiability problem and a consequence of its NP-completeness, *Discrete Applied Mathematics* **52,** 3 (1994) 233–252. ⇒177

[9] A. Mansfield, Determining the thickness of graphs is NP-hard, *Proc. Math. Cambridge Phil. Soc.*, **39** (1983) 9–23. ⇒177

Received: October 31, 2019 • Revised: December 12, 2019

Acta Univ. Sapientiae, Informatica **11**, 2 (2019) 184–205

DOI: 10.2478/ausi-2019-0013

On some $L(2,1)$-coloring parameters of certain graph classes

G. ANJALI
Department of Mathematics
CHRIST (Deemed to be University)
Bangalore, INDIA.
email: anjali.g@maths.christuniversity.in

N. K. SUDEV
Department of Mathematics
CHRIST (Deemed to be University)
Bangalore, INDIA.
email: sudev.nk@christuniversity.in

Abstract. Graph coloring can be considered as a random experiment with the color of a randomly selected vertex as the random variable. In this paper, we consider the $L(2,1)$-coloring of G as the random experiment and we discuss the concept of two fundamental statistical parameters – mean and variance – with respect to the $L(2,1)$-coloring of certain fundamental graph classes.

1 Introduction

For all terms and definitions, not defined specifically in this paper, we refer to [1, 5, 13, 14]. Moreover, for notions and norms in graph colouring, see [2, 6, 8]. Unless mentioned otherwise, all graphs considered here are undirected, simple, finite and connected.

Graph coloring is an assignment of colors or labels or weights to the elements of the graph. A *vertex coloring* of a graph is a function $c : V(G) \to$

Computing Classification System 1998: G.2.2
Mathematics Subject Classification 2010: 05C15, 05C38.
Key words and phrases: $L(2,1)$-coloring, L_1^--chromatic mean, L_1^--chromatic variance, L_1^+-chromatic mean, L_1^+-chromatic variance.

$\mathcal{C} = \{c_1, c_2, c_3, ...c_l\}$, where $\mathcal{C}$ is a set of l distinct colors. Unless mentioned otherwise, by graph coloring, we mean a vertex coloring of G.

A *proper coloring* of a graph G is a coloring such that no two adjacent vertices receive the same color. The *chromatic number* of a graph G, denoted by $\chi(G)$, is the minimum number of colors required in a proper vertex coloring of the graph G.

Note that the color set $\mathcal{C} = \{c_1, c_2, c_3, \dots, c_l\}$ can also be written as $\mathcal{C} = \{1, 2, 3, \dots, \}$. Invoking this representation, we have

Definition 1 [4] The $L(2,1)$*-coloring* of a graph G is a vertex coloring which assigns colors to the vertices of graph G satisfying the following two conditions:

$$\begin{aligned} |c(u) - c(v)| &\geq 2 \text{ if } d(u,v) = 1 \\ |c(u) - c(v)| &\geq 1 \text{ if } d(u,v) = 2 \end{aligned}$$

where u and v are vertices of G.

The *span* of a $L(2,1)$-coloring is its maximum label. The minimum span of a $L(2,1)$-coloring of a graph G is called the $L(2,1)$*-chromatic number* of G. This coloring scheme has significant applications in channel assignment problem and many other fields.

A proper k-coloring of graph G be given by c: $V(G) \to \mathcal{C} = \{c_1, c_2, ...c_k\}$. We denote number of vertices of G receiving the color c_i by $\theta(c_i)$ which is called the *color strength* or *color weight* of the color c_i. The *coloring sum* with respect to a given color set $\mathcal{C}$ of G is defined as $\omega_{\mathcal{C}}(G) = \sum_{i=1}^{k} i\theta(c_i)$ (see [7]).

Recently, some studies have been done by treating graph coloring as a random experiment (see [12, 3, 11, 10, 9]) and the color of an arbitrarily chosen vertex of G as the corresponding discrete random variable X. Then, the *probability mass function* (p.m.f) of this discrete random variable X is defined as

$$f(i) = \begin{cases} \frac{\theta(c_i)}{|V(G)|} & \text{if } i = 1, 2, ...k, \\ 0 & elsewhere. \end{cases}$$

where $\theta(c_i)$ is the cardinality of the color class of G with respect to the color c_i (c.f. [12]). If the context is clear, this p.m.f is referred as the p.m.f of G.

For mean and variance, we use the standard notation μ and σ. Therefore, for a graph G with color set $\mathcal{C}$, the *coloring mean* is defined as

$$\mu_{\mathcal{C}}(G) = \frac{\sum\limits_{i=1}^{k} i\theta(c_i)}{\sum\limits_{i=1}^{k} \theta(c_i)}$$

and the *coloring variance* is defined as

$$\sigma^2_{\mathcal{C}}(G) = \frac{\sum\limits_{i=1}^{k} i^2\theta(c_i)}{\sum\limits_{i=1}^{k} \theta(c_i)} - \left(\frac{\sum\limits_{i=1}^{k} i\theta(c_i)}{\sum\limits_{i=1}^{k} \theta(c_i)}\right)^2$$

In general, the r-th moment is given by

$$\mu_{\mathcal{C}^r}(G) = \frac{\sum\limits_{i=1}^{k} i^r\theta(c_i)}{\sum\limits_{i=1}^{k} \theta(c_i)}$$

where r is any positive integer. If context is clear, we say that $\mu_{\mathcal{C}}(G)$ and $\sigma^2_{\mathcal{C}}(G)$ are the chromatic mean and variance of G.

Motivated by the above studies, in this paper, we extend the notions of chromatic mean and variance to $L(2,1)$-coloring of graphs.

2 Discussion and new results

Throughout this discussion, we denote the $L(2,1)$-color set of G with the minimum possible color by $\mathcal{C}(G)$. In view of this convention, we have the following definitions:

Definition 2 Let $\mathcal{C} = \{c_1, c_2, ...c_l\}$ be the color set corresponding to an $L(2,1)$-coloring c of a given graph G. The coloring mean corresponding to the $L(2,1)$- coloring having minimum chromatic sum is called L_1^--*chromatic mean* of G and is denoted by $\mu_{\mathcal{C}_-}(G)$.

Definition 3 Let $\mathcal{C} = \{c_1, c_2, ...c_l\}$ be the color set corresponding to an $L(2,1)$- coloring c of a given graph G. The coloring variance corresponding to the $L(2,1)$- coloring having minimum chromatic sum is called L_1^--*chromatic variance* of G and is denoted by $\sigma^2_{\mathcal{C}_-}(G)$.

Definition 4 Let $\mathcal{C} = \{c_1, c_2, ...c_l\}$ be the color set corresponding to an $L(2,1)$- coloring c of a given graph G. The coloring mean corresponding to the $L(2,1)$- coloring having maximum chromatic sum is called L_1^+-*chromatic mean* of G and is denoted by $\mu_{\mathcal{C}_+}(G)$.

Definition 5 Let $\mathcal{C} = \{c_1, c_2, ...c_l\}$ be the color set corresponding to an $L(2,1)$- coloring c of a given graph G. The coloring variance corresponding to the $L(2,1)$- coloring having maximum chromatic sum is called L_1^+-*chromatic variance* of G and is denoted by $\sigma^2_{\mathcal{C}_+}(G)$.

In view of the above notions, the chromatic mean and variance corresponding to L_1^- and L_1^+ coloring of complete graphs is discussed below:

Theorem 6 *For a complete graph* K_n*, the coloring parameters,* L_1^-*-chromatic mean and variance and* L_1^+*-chromatic mean and variance are given by*

$$\mu_{\mathcal{C}_-}(K_n) = \mu_{\mathcal{C}_+}(K_n) = n$$

$$\sigma^2_{\mathcal{C}_-}(K_n) = \sigma^2_{\mathcal{C}_+}(K_n) = \frac{n^2-1}{3}$$

Proof. In a complete graph, each vertex receives distinct color and color difference between any two vertices is at least 2. Therefore, we need at least $(2n-1)$ colors say, $\{c_1, c_3, c_5, ...c_{2n-1}\}$, for coloring the vertices of K_n. We cannot use the colors $\{c_2, c_4, ...c_{2n}\}$ by the protocol of $L(2,1)$- coloring. For illustration, see Figure 1. Hence, the corresponding $p.m.f$ is given by

$$f(i) = \begin{cases} \frac{1}{n} & \text{for } i = 1,3,5,...(2n-1), \\ 0 & \text{elsewhere.} \end{cases}$$

Here, we observe that the minimum and maximum coloring sum remains the same. Therefore,

$$\begin{aligned} \mu_{\mathcal{C}_-}(K_n) = \mu_{\mathcal{C}_+}(K_n) &= \frac{1+3+5+....+(2n-1)}{n} \\ &= \frac{1}{n}(n^2) \\ &= n \end{aligned}$$

$$\begin{aligned}\sigma^2_{\mathcal{C}_-}(K_n) = \sigma^2_{\mathcal{C}_+}(K_n) &= \frac{1^2+3^2+5^2+....+(2n-1)^2}{n} - n^2\\ &= \frac{1}{n}\frac{n(2n-1)(2n+1)}{3} - n^2\\ &= \frac{n^2-1}{3}\end{aligned}$$

□

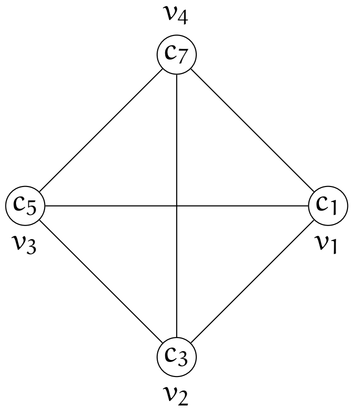


Figure 1

Theorem 7 *For path P_n of length $n \equiv 1, 2 \pmod 3$, The L_1^--chromatic mean is*

$$\mu_{\mathcal{C}_-}(P_n) = \frac{3n-2}{n}$$

and their L_1^--chromatic variance is given by

$$\sigma^2_{\mathcal{C}_-}(P_n) = \begin{cases} \frac{8n^2+4n-12}{3n^2} & \text{if } n \equiv 1 \pmod 3\\ \frac{8n^2-4n-12}{3n^2} & \text{if } n \equiv 2 \pmod 3\end{cases}$$

Also, for $n \equiv 0 \pmod 3$, the L_1^--chromatic mean for path P_n is

$$\mu_{\mathcal{C}_-}(P_n) = \begin{cases} 3 & \text{if } n \equiv 0 \pmod 5,\\ \frac{3n-2}{n} & \text{if } n \equiv 1,2,3 \pmod 5\\ \frac{3n-1}{n} & \text{if } n \equiv 4 \pmod 5\end{cases}$$

and it's L_1^-*-chromatic variance is given by*

$$\sigma^2_{\mathcal{C}_-}(P_n) = \begin{cases} 2 & \textit{if } n \equiv 0 \pmod 5 \\ \dfrac{2n^2+2n-4}{n^2} & \textit{if } n \equiv 1 \pmod 5 \\ \dfrac{2n^2-4}{n^2} & \textit{if } n \equiv 2,3 \pmod 5 \\ \dfrac{2n^2+n-1}{n^2} & \textit{if } n \equiv 4 \pmod 5 \end{cases}$$

Proof. Note that according to the $L(2,1)$- coloring protocol, any three consecutive vertices of P_n must receive distinct colors. $L(2,1)$- chromatic number of P_n is 3, thus we have the color set as $\{c_1, c_2, c_3, c_4, c_5\}$. Now let us consider each case separately.
Case 1: When $n \equiv 1 \pmod 3$, we observe that $(\frac{n+2}{3})$ vertices receive the color c_1, $(\frac{n-1}{3})$ vertices each receive color c_3 and c_5. In accordance with $L(2,1)$-coloring protocol, c_2 and c_4 cannot be assigned to any vertex. Then, the corresponding $p.m.f$ is given by

$$f(i) = \begin{cases} \dfrac{n+2}{3n} & \text{if } i = 1, \\ \dfrac{n-1}{3n} & \text{if } i = 3, 5, \\ 0 & \text{elsewhere.} \end{cases}$$

Therefore, the L_1^--chromatic mean $= (1)\frac{n+2}{3n} + (3+5)\frac{n-1}{3n} = \frac{3n-2}{n}$ and variance $= (1^2)\frac{n+2}{3n} + (3^2+5^2)\frac{n-1}{3n} - (\frac{3n-2}{n})^2 = \frac{8n^2+4n-12}{3n^2}$ (refer to Figure 2).

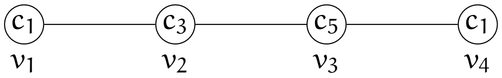


Figure 2

Case 2: When $n \equiv 2 \pmod 3$, we observe that $(\frac{n+1}{3})$ vertices each receive the color c_1 and c_3, $(\frac{n-2}{3})$ vertices receive color c_5. In accordance with $L(2,1)$-coloring protocol, c_2 and c_4 cannot be assigned to any vertex. Then, the corresponding $\mathrm{p.m.f}$ is given by

$$f(i) = \begin{cases} \dfrac{n+1}{3n} & \text{if } i = 1, 3, \\ \dfrac{n-2}{3n} & \text{if } i = 5, \\ 0 & \text{elsewhere.} \end{cases}$$

Then, the L_1^--chromatic mean $= (1+3)\frac{n+1}{3n} + (5)\frac{n-2}{3n} = \frac{3n-2}{n}$ and variance $= (1^2+3^2)\frac{n+1}{3n} + (5^2)\frac{n-2}{3n} - (\frac{3n-2}{n})^2 = \frac{8n^2-4n-12}{3n^2}$ (refer to Figure 3).

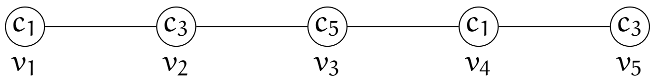


Figure 3

Case 3: When $n \equiv 0 \pmod 5$, each color c_1, c_2, c_3, c_4 and c_5 is given to $(\frac{n}{5})$ vertices. Then, the corresponding $\mathrm{p.m.f}$ is given by

$$f(i) = \begin{cases} \frac{1}{5} & \text{if } i = 1, 2, 3, 4, 5, \\ 0 & \text{elsewhere.} \end{cases}$$

The L_1^--chromatic mean $= \sum\limits_{i=1}^{5}(i)\frac{1}{5} = \frac{15}{5} = 3$ and

variance $= \sum\limits_{i=1}^{5}(i^2)\frac{1}{5} - (3^2) = 11 - 9 = 2$ (refer to Figure 4).

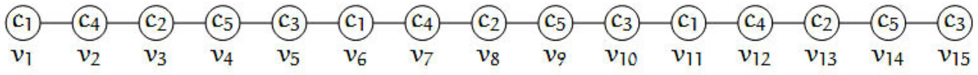
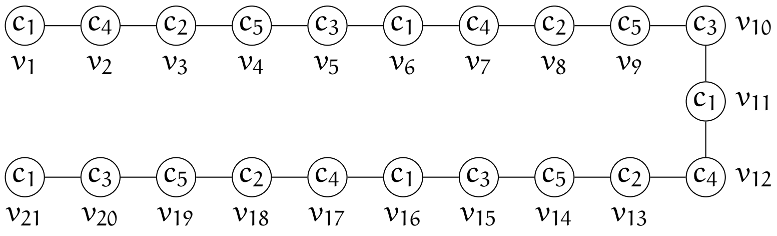


Figure 4

Case 4: When $n \equiv 1 \pmod 5$, we shall see that $(\frac{n+4}{5})$ vertices receive color c_1, and $(\frac{n-1}{5})$ vertices each receive color c_2, c_3, c_4 and c_5. Then, the $\mathrm{p.m.f}$ is given by

$$f(i) = \begin{cases} \dfrac{n+4}{5n} & \text{if } i = 1 \\ \dfrac{n-1}{5n} & \text{if } i = 2, 3, 4, 5, \\ 0 & \text{elsewhere.} \end{cases}$$

The L_1^--chromatic mean $= (1)\frac{n+4}{5n} + (2+3+4+5)\frac{n-1}{5n} = \frac{15n-10}{5n} = \frac{3n-2}{n}$ and variance $= (1^2)\frac{n+4}{5n} + (2^2+3^2+4^2+5^2)\frac{n-1}{5n} - (\frac{3n-2}{n})^2 = \frac{2n^2+2n-4}{n^2}$ (refer to Figure 5).



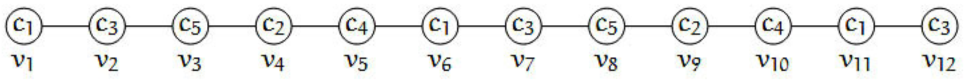
c_1 c_4 c_2 c_5 c_3 c_1 c_4 c_2 c_5 c_3
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9 v_{10}
c_1 v_{11}
c_4 v_{12}
c_1 c_3 c_5 c_2 c_4 c_1 c_3 c_5 c_2
v_{21} v_{20} v_{19} v_{18} v_{17} v_{16} v_{15} v_{14} v_{13}

Figure 5

Case 5: When $n \equiv 2 \pmod 5$, we shall give $(\frac{n+3}{5})$ vertices each color c_1 and c_3; $(\frac{n-2}{5})$ vertices each color c_2, c_4 and c_5. Then, the $p.m.f$ is given by

$$f(i) = \begin{cases} \frac{n+3}{5n} & \text{if } i = 1,3, \\ \frac{n-2}{5n} & \text{if } i = 2,4,5 \\ 0 & \text{elsewhere.} \end{cases}$$

The L_1^--chromatic mean $= (1+3)\frac{n+3}{5n} + (2+4+5)\frac{n-2}{5n} = \frac{15n-10}{5n} = \frac{3n-2}{n}$ and variance $= (1^2+3^2)\frac{n+3}{5n} + (2^2+4^2+5^2)\frac{n-2}{5n} - (\frac{3n-1}{n})^2 = \frac{2n^2-4}{n^2}$ (refer to Figure 6).



c_1 c_3 c_5 c_2 c_4 c_1 c_3 c_5 c_2 c_4 c_1 c_3
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9 v_{10} v_{11} v_{12}

Figure 6

Case 6: When $n \equiv 3 \pmod 5$, we observe that each $(\frac{n+2}{5})$ vertices receive the color c_1, c_4 and c_2; and each $(\frac{n-3}{5})$ vertices receive the color c_3 and c_5.

Then, the $\mathsf{p.m.f}$ is given by

$$f(i) = \begin{cases} \dfrac{n+2}{5n} & \text{if } i = 1,4,2 \\ \dfrac{n-3}{5n} & \text{if } i = 3,5, \\ 0 & \text{elsewhere.} \end{cases}$$

The L_1^--chromatic mean $= (1+2+4)\frac{n+2}{5n} + (3+5)\frac{n-3}{5n} = \frac{15n-10}{5n} = \frac{3n-2}{n}$ and variance $= (1^2+2^2+4^2)\frac{n+2}{5n}+(3^2++5^2)\frac{n-3}{5n}-(\frac{3n-2}{n})^2 = \frac{2n^2-4}{n^2}$ (refer to Figure 7).

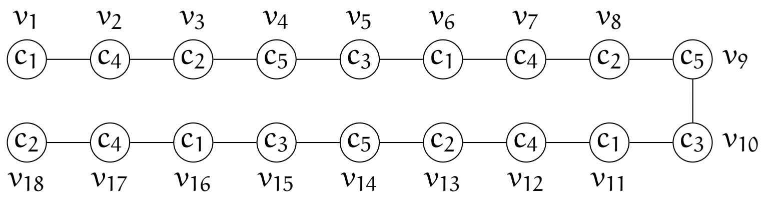


Figure 7

Case 7: When $n \equiv 4 \pmod 5$, we observe that $(\frac{n-4}{5})$vertices receive color c_4, $(\frac{n+1}{5})$ vertices each receives color c_1, c_2, c_3 and c_5. Then, the $\mathsf{p.m.f}$ is given by

$$f(i) = \begin{cases} \dfrac{n+1}{5n} & \text{if } i = 1,2,3,5 \\ \dfrac{n-4}{5n} & \text{if } i = 4, \\ 0 & \text{elsewhere.} \end{cases}$$

The L_1^--chromatic mean $= (1+2+3+5)\frac{n+1}{5n} + (4)\frac{n-4}{5n} = \frac{15n-5}{5n} = \frac{3n-1}{n}$ and variance $= (1^2+2^2+3^2+5^2)\frac{n+1}{5n}+(4^2)\frac{n-4}{5n}-(\frac{3n-1}{5n})^2 = \frac{2n^2+n-1}{n^2}$ (refer to Figure 8).

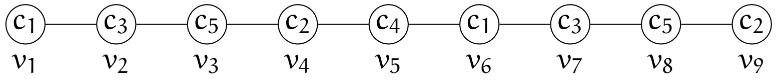


Figure 8

□

Theorem 8 *The* L_1^+*-chromatic mean of path* P_n *of length* $n \equiv 1, 2 \pmod 3$ *is*

$$\mu_{\mathcal{C}_+}(P_n) = \frac{3n+2}{n}$$

and their L_1^+*-chromatic variance is given by*

$$\sigma^2_{\mathcal{C}_+}(P_n) = \begin{cases} \dfrac{8n^2+4n-12}{3n^2} & \textit{if } n \equiv 1 \pmod 3 \\ \dfrac{8n^2-4n-12}{3n^2} & \textit{if } n \equiv 2 \pmod 3. \end{cases}$$

Also, the L_1^+*-chromatic mean for path* P_n *of length* $n \equiv 0 \pmod 3$ *is*

$$\mu_{\mathcal{C}_+}(P_n) = \begin{cases} 3 & \textit{if } n \equiv 0 \pmod 5 \\ \dfrac{3n+1}{n} & \textit{if } n \equiv 1 \pmod 5 \\ \dfrac{3n+2}{n} & \textit{if } n \equiv 2,3,4 \pmod 5 \\ 0 & \textit{elsewhere.} \end{cases}$$

and it's L_1^+*-chromatic variance is given by*

$$\sigma^2_{\mathcal{C}_+}(P_n) = \begin{cases} 2 & \textit{if } n \equiv 0 \pmod 5 \\ \dfrac{2n^2-n-1}{n^2} & \textit{if } n \equiv 1 \pmod 5 \\ \dfrac{2n^2-4}{n^2} & \textit{if } n \equiv 2,3 \pmod 5 \\ \dfrac{2n^2-2n-4}{n^2} & \textit{if } n \equiv 4 \pmod 5 \\ 0 & \textit{elsewhere.} \end{cases}$$

Proof. In accordance with $L(2,1)$- coloring protocol, any three consecutive vertices of P_n must receive distinct colors. $L(2,1)$- chromatic number of P_n is 3 and the corresponding color set is $\{c_1, c_2, c_3, c_4, c_5\}$. Considering each case separately,
Case 1: When $n \equiv 1 \pmod 3$, we shall give color c_5 to $(\frac{n+2}{3})$ vertices and color c_1 and c_3 to $(\frac{n-1}{3})$ vertices. c_2 and c_4 cannot be assigned to any vertex according to the $L(2,1)$- coloring protocol. Then, the corresponding $p.m.f$ is

given by

$$f(i)=\begin{cases}\dfrac{n-1}{3n} & \text{if } i=1,3,\\ \dfrac{n+2}{3n} & \text{if } i=5,\\ 0 & \text{elsewhere.}\end{cases}$$

Then, the L_1^+-chromatic mean $= (1+3)\frac{n-1}{3n} + (5)\frac{n+2}{3n} = \frac{3n+2}{n}$ and variance $= (1^2+3^2)\frac{n-1}{3n} + (5^2)\frac{n+2}{3n} - (\frac{3n+2}{n})^2 = \frac{8n^2+4n-12}{3n^2}$ (refer to Figure 9).

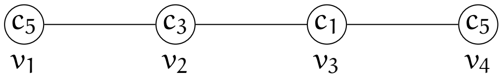


Figure 9

Case 2: When $n \equiv 2 \pmod 3$, we shall give $(\frac{n-2}{3})$ vertices color c_1 and $(\frac{n+1}{3}$ vertices each receive color c_3 and c_5. Then, the p.m.f is given by

$$f(i)=\begin{cases}\frac{n-2}{3n} & \text{if } i=1,\\ \frac{n+1}{3n} & \text{if } i=3,5,\\ 0 & \text{elsewhere.}\end{cases}$$

Then, the L_1^+-chromatic mean $= (1)\frac{n-2}{3n} + (3+5)\frac{n+1}{3n} = \frac{3n+2}{n}$ and variance $= (1^2)\frac{n-2}{3n} + (3^2+5^2)\frac{n+1}{3n} - (\frac{3n+2}{n})^2 = \frac{8n^2-4n-12}{3n^2}$ (refer to Figure 10).

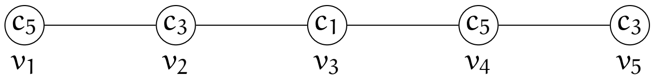


Figure 10

When n ≡ 0 (mod 3), the p.m.f is given by

Case 3: When $n \equiv 0 \pmod 5$, each color c_1, c_2, c_3, c_4 and c_5 is given to $(\frac{n}{5})$ vertices. Then, the corresponding p.m.f is given by

$$f(i)=\begin{cases}\dfrac{1}{5} & \text{if } i=1,2,3,4,5,\\ 0 & \text{elsewhere.}\end{cases}$$

The L_1^+-chromatic mean $= \sum\limits_{i=1}^{5}(i)\frac{1}{5} = \frac{15}{5} = 3$ and

variance $= \sum\limits_{i=1}^{5}(i^2)\frac{1}{5} - (3^2) = 11 - 9 = 2$ (refer to Figure 11).

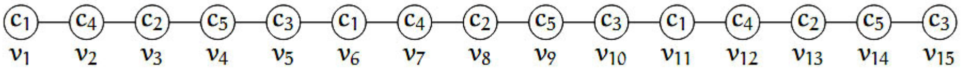

c_1 c_4 c_2 c_5 c_3 c_1 c_4 c_2 c_5 c_3 c_1 c_4 c_2 c_5 c_3
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9 v_{10} v_{11} v_{12} v_{13} v_{14} v_{15}

Figure 11

Case 4: When $n \equiv 1 \pmod 5$, we observe that $(\frac{n+4}{5})$ vertices receive the color c_4 and $(\frac{n-1}{5})$ vertices each receive c_1, c_2, c_4 and c_5. Then, the $p.m.f$ is given by

$$f(i) = \begin{cases} \dfrac{n+4}{5n} & \text{if } i = 4 \\ \dfrac{n-1}{5n} & \text{if } i = 1,2,3,5, \\ 0 & \text{elsewhere.} \end{cases}$$

The L_1^+-chromatic mean $= (4)\frac{n+4}{5n} + (1+2+3+5)\frac{n-1}{5n} = \frac{15n+5}{5n} = \frac{3n+1}{n}$ and variance $= (4^2)\frac{n+4}{5n} + (1^2+2^2+3^2+5^2)\frac{n-1}{5n} - (3)^2 = \frac{2n^2-n-1}{n^2}$ (refer to Figure 12).

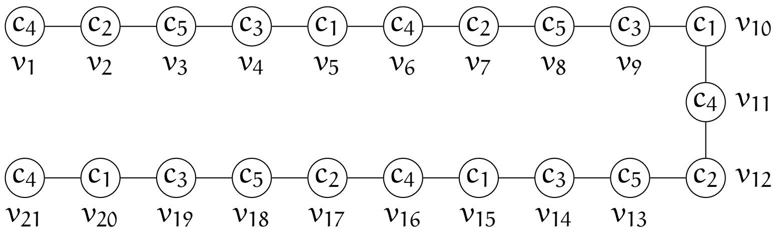

c_4 c_2 c_5 c_3 c_1 c_4 c_2 c_5 c_3 c_1
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9 v_{10}
c_4 v_{11}
c_4 c_1 c_3 c_5 c_2 c_4 c_1 c_3 c_5 c_2 v_{12}
v_{21} v_{20} v_{19} v_{18} v_{17} v_{16} v_{15} v_{14} v_{13}

Figure 12

Case 5: When $n \equiv 2 \pmod 5$, we observe that $(\frac{n+3}{5})$ vertices each receive

c_3, c_5 and $(\frac{n-2}{5})$ vertices each receive c_1, c_2 and c_4. The $p.m.f$ is given by

$$f(i) = \begin{cases} \dfrac{n+3}{5n} & \text{if } i = 3, 5 \\ \dfrac{n-2}{5n} & \text{if } i = 1, 2, 4, \\ 0 & \text{elsewhere.} \end{cases}$$

The L_1^+-chromatic mean $= (3+5)\frac{n+3}{5n} + (1+2+4)\frac{n-2}{5n} = \frac{15n+10}{5n} = \frac{3n+2}{n}$ and variance $= (3^2+5^2)\frac{n+3}{5n} + (1^2+2^2+4^2)\frac{n-2}{5n} - (\frac{3n+2}{n})^2 = \frac{2n^2-4}{n^2}$ (refer to Figure 13).

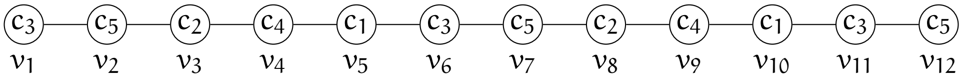


Figure 13

Case 6: When $n \equiv 3 \pmod 5$, we shall give color c_2, c_4 and c_5 to each $(\frac{n+2}{5})$ vertices and color c_1 and c_3 to each $(\frac{n-3}{5})$ vertices. Then, the corresponding $p.m.f$ is given by

$$f(i) = \begin{cases} \dfrac{n+2}{5n} & \text{if } i = 2, 4, 5 \\ \dfrac{n-3}{5n} & \text{if } i = 1, 3, \\ 0 & \text{elsewhere.} \end{cases}$$

The L_1^+-chromatic mean $= (2+4+5)\frac{n+2}{5n} + (1+3)\frac{n-3}{5n} = \frac{15n+10}{5n} = \frac{3n+2}{n}$ and variance $= (2^2+4^2+5^2)\frac{n+2}{5n} + (1^2+3^2)\frac{n-3}{5n} - (\frac{3n+2}{n})^2 = \frac{2n^2-4}{n^2}$ (refer to Figure 14).

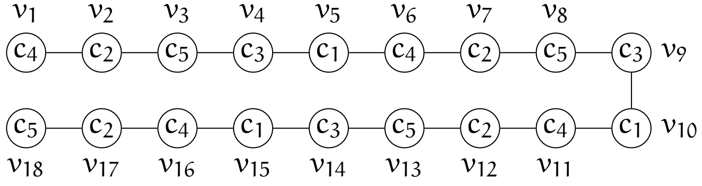


Figure 14

Case 7: When $n \equiv 4 \pmod 5$, we give c_1 to $(\frac{n-4}{5})$ vertices and each color c_2, c_3, c_4, c_5 to $(\frac{n+1}{5})$ vertices. Then, the corresponding p.m.f is given by

$$f(i) = \begin{cases} \dfrac{n+1}{5n} & \text{if } i = 2,3,4,5 \\ \dfrac{n-4}{5n} & \text{if } i = 1, \\ 0 & \text{elsewhere.} \end{cases}$$

The L_1^+-chromatic mean $= (2+3+4+5)\frac{n+1}{5n} + (1)\frac{n-4}{5n} = \frac{15n+10}{5n} = \frac{3n+2}{n}$ and variance $= (2^2+3^2+4^2+5^2)\frac{n+1}{5n} + (1^2)\frac{n-4}{5n} - (\frac{3n+2}{n})^2 = \frac{2n^2-2n-4}{n^2}$ (refer to Figure 15).

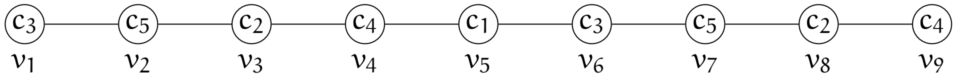


Figure 15

□

Next our aim is to find L_1^--chromatic mean of cycles. Consider C_3 and C_6 and their color set $\{c_1, c_3, c_5\}$, their p.m.f is given by

$$f(i) = \begin{cases} \dfrac{1}{3} & \text{if } i = 1,3,5, \\ 0 & \text{elsewhere.} \end{cases}$$

The L_1^--chromatic mean $= (1+3+5)\frac{1}{3} = \frac{9}{3} = 3$ and
variance $= (1^2+3^2+5^2)\frac{1}{3} - (3^2) = \frac{35}{3} - 9 = \frac{8}{3}$.
Therefore, for C_3 and C_6 L_1^--chromatic mean is 3 and L_1^--chromatic variance is $\frac{8}{3}$.

Theorem 9 *The* L_1^-*-chromatic mean of cycle* C_n *where* $n \neq 3, 6$ *is*

$$\mu_{\mathcal{C}_-}(C_n) = 3$$

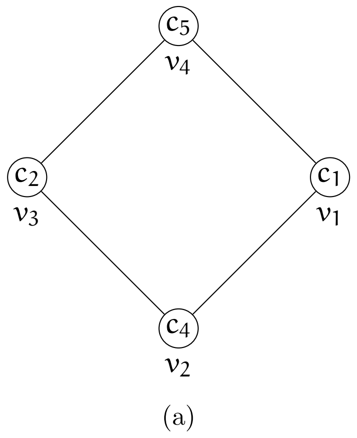
and L_1^-*-chromatic variance for* C_n *where* $n \neq 3, 6$ *is given by*

$$\sigma^2_{\mathcal{C}_-}(C_n) = \begin{cases} \frac{5}{2} & \textit{if } n \equiv 0 \text{ mod } 4) \\ \frac{5n-5}{2n} & \textit{if } n \equiv 1 \text{ (mod } 4) \\ \frac{5n-10}{2n} & \textit{if } n \equiv 2 \text{ (mod } 4) \\ \frac{5n+1}{2n} & \textit{if } n \equiv 3 \text{ (mod } 4) \end{cases}$$

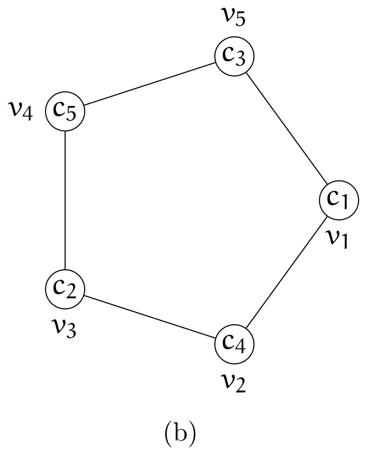
Proof. From the definition of $L(2,1)$- coloring, any three consecutive vertices of C_n must receive distinct colors. Chromatic number of C_n is 5 and the color classes used are c_1, c_2, c_3, c_4, c_5. Now let us consider each case separately.
Case 1: When $n \equiv 0$ (mod 4),each color c_1, c_2, c_4 and c_5 is received by $(\frac{n}{4})$ vertices. Hence, the $p.m.f$ is given by

$$f(i) = \begin{cases} \frac{1}{4} & \text{if } i = 1, 2, 4, 5, \\ 0 & \text{elsewhere.} \end{cases}$$

The L_1^--chromatic mean $= (1+2+4+5)\frac{1}{4} = \frac{12}{4} = 3$ and
variance $= (1^2 + 2^2 + 4^2 + 5^2)\frac{1}{4} - (3^2) = \frac{46}{4} - 9 = \frac{5}{2}$ (refer to Figure 16a).



(a)



(b)

Figure 16

Case 2: When $n \equiv 1 \pmod 4$, c_3 is given to the last vertex i.e. v_n and remaining $(\frac{n-1}{4})$ vertices each receive c_1, c_2, c_4 and c_5. Then, the $p.m.f$ is given by

$$f(i) = \begin{cases} \dfrac{n-1}{4n} & \text{if } i = 1,2,4,5 \\ \dfrac{1}{n} & \text{if } i = 3, \\ 0 & \text{elsewhere.} \end{cases}$$

The L_1^--chromatic mean $= (1+2+4+5)\frac{n-1}{4n} + (3)\frac{1}{n} = 3$ and
variance $= (1^2+2^2+4^2+5^2)\frac{n-1}{4n} + (3^2)\frac{1}{n} - (3)^2 = \frac{5n-5}{2n}$ (refer to Figure 16b).
Case 3: When $n \equiv 2 \pmod 4$, we observe that two vertices receive c_3 and $(\frac{n-2}{4})$ vertices each receive c_1, c_2, c_4 and c_5. Then, the $p.m.f$ is given by

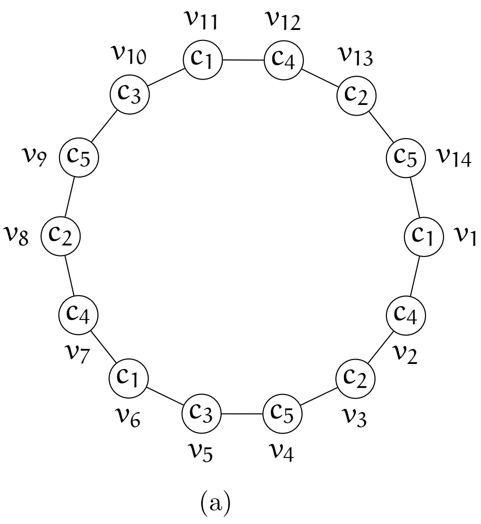
$$f(i) = \begin{cases} \dfrac{n-2}{4n} & \text{if } i = 1,2,4,5 \\ \dfrac{2}{n} & \text{if } i = 3, \\ 0 & \text{elsewhere.} \end{cases}$$

The L_1^--chromatic mean $= (1+2+4+5)\frac{n-2}{4n} + (3)\frac{2}{n} = 3$ and
variance $= (1^2+2^2+4^2+5^2)\frac{n-2}{4n} + (3^2)\frac{2}{n} - (3)^2 = \frac{5n-10}{2n}$ (refer to Figure 17a).

Case 4: When $n \equiv 3 \pmod 4$,each set of $(\frac{n+1}{4})$ vertices receive c_1 and c_5; each set of $(\frac{n-3}{4})$ vertices receive c_2 and c_4; and one vertex receives c_3. Then, the corresponding $p.m.f$ is given by

$$f(i) = \begin{cases} \dfrac{n+1}{4n} & \text{if } i = 1,5 \\ \dfrac{n-3}{4n} & \text{if } i = 2,4 \\ \dfrac{1}{n} & \text{if } i = 3, \\ 0 & \text{elsewhere.} \end{cases}$$

The L_1^--chromatic mean $= (1+5)\frac{n+1}{4n} + (2+4)\frac{n-3}{4n} + (3)\frac{1}{n} = 3$ and
variance $= (1^2+5^2)\frac{n+1}{4n} + (2^2+4^2)\frac{n-3}{4n} + (3^2)\frac{1}{n} - (3)^2 = \frac{5n+1}{2n}$ (refer to Figure 17b). $\square$



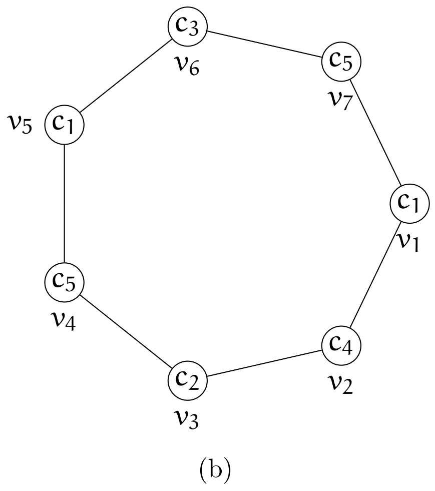


Figure 17

Theorem 10 *The* L_1^+*-chromatic mean of cycle of length* $n \equiv 0 \pmod 3$ *is*

$$\mu_{\mathcal{C}_+}(C_n) = 3$$

and L_1^+*-chromatic variance by*

$$\sigma^2_{\mathcal{C}_+}(C_n) = \frac{8}{3}$$

Proof. In case of $n \equiv 1{,}2 \pmod 3$, L_1^- and L_1^+-chromatic mean are same and so is the case of L_1^- and L_1^+-chromatic variance. Therefore, we just consider the cycle of length $n \equiv 0 \pmod 3$. Here, $(\frac{n}{3})$ vertices each receive color c_1, c_3 and $c_5.c_2$ and c_4 are not received by any vertex of graph G. For illustration, see Figure 18a. The corresponding $p.m.f$ for L_1^+ coloring is given by

$$f(i) = \begin{cases} \frac{1}{3}, & \text{for i = 1, 3, 5} \\ 0 & \text{elsewhere} \end{cases}$$

The L_1^+-chromatic mean $= (1{+}3{+}5)\frac{1}{3} = 3$ and variance $= (1^2{+}3^2{+}5^2)\frac{1}{3} - (3)^2 = \frac{8}{3}$.

□

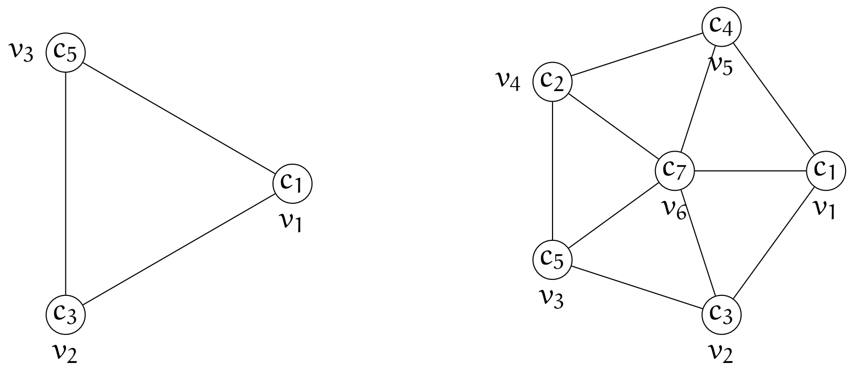


Figure 18

Theorem 11 *For wheel graphs having* n *vertices, where* $n \geq 6$*, mean and variance for* L_1^- *and* L_1^+ *coloring are given by*

$$\mu_{\mathcal{C}_-}(W_n) = \mu_{\mathcal{C}_+}(W_n) = \frac{n^2+n+2}{2n}$$

and

$$\sigma^2_{\mathcal{C}_-}(W_n) = \sigma^2_{\mathcal{C}_+}(W_n) = \frac{n^4+11n^2-12}{12n^2}$$

Proof. The diameter of wheel graph is 2. Also, the central vertex is adjacent to all the other vertices. Hence, we need (n+1) colors. We give the color c_{n+1} to the central vertex and remaining colors to the other vertices of G. Its $p.m.f$ is given by

$$f(i) = \begin{cases} \frac{1}{n}, & \text{if } i = 1, 2, \ldots(n-1), (n+1) \\ 0 & \text{elsewhere} \end{cases}$$

Therefore, L_1^- and L_1^+-chromatic mean $= (1+2+\ldots n-1+n+1)\frac{1}{n} = \frac{n^2+n+2}{2n}$. L_1^- and L_1^+ chromatic variance $= (1^2+2^2+\ldots(n-1)^2+(n+1)^2\frac{1}{n} - (\frac{n^2+n+2}{2n})^2 = \frac{n^4+11n^2-12}{12n^2}$ (refer to Figure 18b). □

Theorem 12 *For helm graphs having* $2n+1$ *vertices, where* $n \geq 7$*,* L_1^-*-chromatic mean is given by*

$$\mu_{\mathcal{C}_-}(H_n) = \frac{n^2+5n+28}{4n+2}$$

and L_1^-*-chromatic variance is given by*

$$\sigma^2_{\mathcal{C}_-}(H_n) = \frac{2n^3+9n^2+31n+198}{6(2n+1)}$$

Proof. We need $n+2$ colors to color the vertices of helm graph. The wheel graph induced from the given helm graph is colored as discussed in the previous theorem. Among the remaining n vertices, $n-4$ vertices receive c_1, 2 vertices receive C_2, and c_3, c_4 is given to one vertex each. For illustration, see Figure 19a. The corresponding $p.m.f$ is given by:

$$f(i) = \begin{cases} \frac{n-3}{2n+1} & \text{if } i = 1 \\ \frac{3}{2n+1} & \text{if } i = 2 \\ \frac{2}{2n+1} & \text{if } i = 3,4 \\ \frac{1}{2n+1} & \text{if } i = 5,6,7,...n,(n+2) \\ 0 & \text{elsewhere.} \end{cases}$$

L_1^--chromatic mean $= 1\frac{n-3}{2n+1}+(2)\frac{3}{2n+1}+(3+4)\frac{2}{2n+1}+(5+6+...n+(n+2))\frac{1}{2n+1} = \frac{n^2+5n+28}{4n+2}$ and variance $= (1^2)\frac{n-3}{2n+1} + (2^2)\frac{3}{2n+1} + (3^2+4^2)\frac{2}{2n+1}(5^2+6^2+..n^2+(n+2)^2)\frac{1}{2n+1} - (\frac{n^2+5n+28}{4n+2})^2 = \frac{2n^3+9n^2+31n+198}{6(2n+1)}$

□

Theorem 13 *For helm graphs having* $2n+1$ *vertices, where* $n \geq 7$, L_1^+*-chromatic mean is given by*

$$\mu_{\mathcal{C}_+}(H_n) = \frac{n^2+2n+2}{2n+1}$$

and L_1^+*-chromatic variance is given by*

$$\sigma^2_{\mathcal{C}_+}(H_n) = \frac{n^4+2n^3+8n^2+13n}{3(2n+1)^2}$$

Proof. We need $n+2$ colors to color the vertices of helm graph. The wheel graph induced from the given helm graph is colored as discussed in Theorem

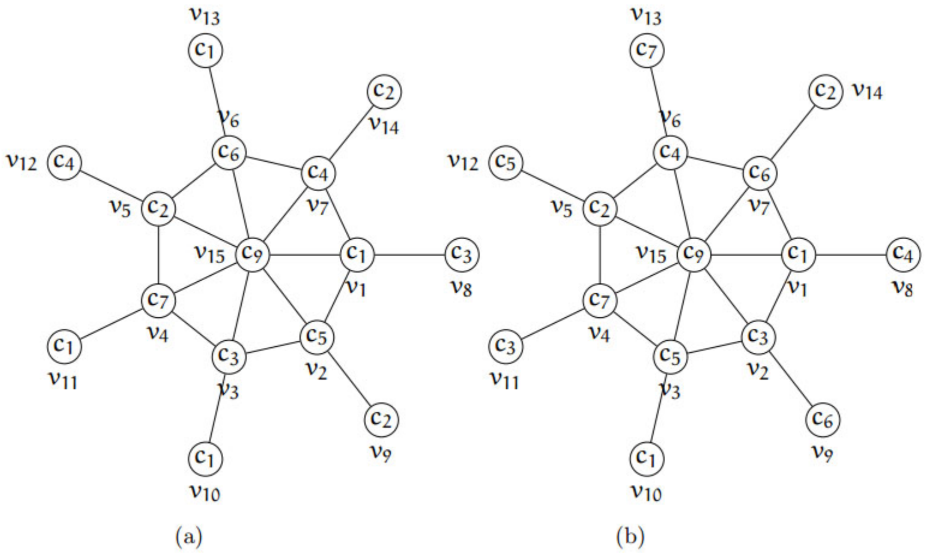


Figure 19

6. And each vertex in the remaining n vertices receive distinct color c_i (where $i = 1, 2, ..n$). The corresponding $p.m.f$ is given by:

$$f(i) = \begin{cases} \dfrac{2}{2n+1} & 1,\ 2,\ \ldots\ n \\ \dfrac{1}{2n+1} & n+2 \end{cases}$$

L_1^+-chromatic mean $= (1+2+..n)\frac{2}{2n+1} + (n+2)\frac{1}{2n+1} = \frac{n^2+2n+2}{2n+1}$ and variance $= (1^2+2^2+...n^2)\frac{2}{2n+1} + (n+2)^2\frac{1}{2n+1} - (\frac{n^2+2n+2}{2n+1})^2 = \frac{n^4+2n^3+8n^2+13n}{3(2n+1)^2}$ (refer to Figure 19b). □

Theorem 14 *For flower graph having* $n+1$ *vertices, where* $n \geq 6$, L_1^- *and* L_1^+*-chromatic mean and variance are given by*

$$\mu_{\mathcal{C}_-}(Fl_n) = \mu_{\mathcal{C}_+}(Fl_n) = \frac{n^2+3n+4}{2n+2}$$

and

$$\sigma^2_{\mathcal{C}_-}(Fl_n) = \sigma^2_{\mathcal{C}_+}(Fl_n) = \frac{n^4+4n^3+17n^2+26n}{12(n+1)^2}$$

Proof. The diameter of flower graph is 2. Thus, each vertex receives distinct color and central vertex is adjacent to all the other vertices. By definition, color difference between central vertex and any other vertex is 2. so we shall give the color c_{n+2} to the central vertex and other vertices receive distinct color c_i, (where $i = 1, 2, ...n$). For illustration, see Figure 20. The corresponding $p.m.f$ is given by

$$f(i) = \begin{cases} \dfrac{1}{n+1}, & \text{if } i = 1, 2, ...n, (n+2) \\ 0 & \text{elsewhere.} \end{cases}$$

Therefore, L_1^- and L_+^+ chromatic mean $= (1 + 2 + ...n + (n+2))\frac{1}{n+1} = \frac{n^2+3n+4}{2n+2}$ and variance $= (1^2 + 2^2 + ...n^2 + (n+2)^2)\frac{1}{n+1} - (\frac{n^2+2n+2}{n+1})^2 = \frac{n^4+4n^3+17n^2+26n}{12(n+1)^2}$

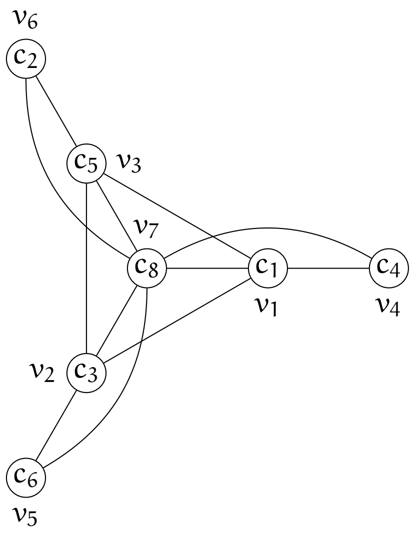


Figure 20

□

3 Conclusion

In this paper, we have introduced the notions of certain coloring means and variances related to $L(2, 1)$-coloring and discussed these parameters in context of some fundamental graph classes. Further investigations are possible in this area, as the above-mentioned parameters can be discussed for many other classes of graphs, graph operations, graph products and known derived graphs. The coloring parameters play vital role in many areas such as network analysis, distribution problems, transportation problems, etc.

References

[1] J. A. Bondy, U. S. R. Murty, *Graph theory.* Springer, New York, 2008. ⇒184

[2] G. Chartrand, P. Zhang, *Chromatic graph theory.* Chapman and Hall/CRC, 2008. ⇒184

[3] K. P. Chithra, E. A. Shiny, N. K. Sudev, On equitable coloring parameters of certain cycle related graphs. *Contemp. Stud. Discrete Math.*, **1,** 1 (2018) 3–15. ⇒185

[4] J. R. Griggs, R. K. Yeh, Labelling graphs with a condition at distance 2. *SIAM J. Discrete Math.*, **5,** 4 (1992) 586–595. ⇒185

[5] F. Harary, *Graph theory*, Narosa Publ., NewDelhi, 2001. ⇒184

[6] T. R. Jensen, B. Toft, *Graph coloring problems.* John Wiley & Sons, 2011. ⇒ 184

[7] J. Kok, N. K. Sudev, K. P. Chithra, Generalised colouring sums of graphs. *Cogent Mathematics*, **3,** 1 (2016) 1140002:1–11. ⇒185

[8] M. Kubale, *Graph colorings*, American Mathematical Soc., 2004. ⇒184

[9] N. K. Sudev, K. P. Chithra, J. Kok, Certain chromatic sums of some cycle-related graph classes. *Discrete Math. Algorithm. Appl.*, **8,** 03 (2016) 1650050:1–25. ⇒ 185

[10] N. K. Sudev, K. P. Chithra, S. Satheesh, J. Kok. A study on the injective coloring parameters of certain graphs. *Discrete Math. Algorithm. Appl.*, **8,** 03 (2016) 1650052 ⇒185

[11] N. K. Sudev, K. P. Chithra, S. Satheesh, J. Kok, On certain parameters of equitable coloring of graphs. *Discrete Math. Algorithm. Appl.*, **9,** 04 (2017) 1750054:1–11. ⇒185

[12] N. K. Sudev, S. Satheesh, K. P. Chithra, J. Kok, On certain colouring parameters of graphs. *Int. J. Math. Combin.*, **3** (2018) 87–98. ⇒185

[13] E. W. Weisstein, *CRC concise encyclopedia of mathematics.* Chapman and Hall/CRC, 2002. ⇒184

[14] D. B. West, *Introduction to graph theory*, volume 2. Prentice Hall of India, New Delhi., 2001. ⇒184

Received: October 22, 2019 • Revised: November 30, 2019

Acta Univ. Sapientiae, Informatica **11,** 2 (2019) 206–214

DOI: 10.2478/ausi-2019-0014

Heuristic method to determine lucky k-polynomials for k-colorable graphs

Johan KOK
CHRIST (Deemed to be a University), Bangalore,
India
email: jacotype@gmail.com

Abstract. The existence of edges is a huge challenge with regards to determining lucky k-polynomials of simple connected graphs in general. In this paper the lucky 3-polynomials of path and cycle graphs of order, $3 \leq n \leq 8$ are presented as the basis for the heuristic method to determine the lucky k-polynomials for k-colorable graphs. The difficulty of adjacency with graphs is illustrated through these elementary graph structures . The results are also illustratively compared with the results for null graphs (edgeless graphs). The paper could serve as a basis for finding recurrence results through innovative methodology.

1 Introduction

For general notation and concepts in graphs see, [1, 2, 6]. It is assumed that the reader is familiar with the concept of graph coloring. Recall that in a proper coloring of G all edges are good i.e. $uv \Leftrightarrow c(u) \neq c(v)$. For any proper coloring $\varphi(G)$ of a graph G the addition of all good edges, if any, is called the chromatic completion of G in respect of $\varphi(G)$. The additional edges are called *chromatic completion edges.* The set of such chromatic completion edges is denoted by, $E_\varphi(G)$. The resultant graph G_φ is called a *chromatic completion graph* of G. See [3] for an introduction to chromatic completion of a graph.

Computing Classification System 1998: G.2.2
Mathematics Subject Classification 2010: 05C15, 05C38, 05C75, 05C85
Key words and phrases: chromatic completion, perfect lucky 3-coloring, lucky 3-polynomial

The *chromatic completion number* of a graph G denoted by, $\zeta(G)$ is the maximum number of good edges that can be added to G over all chromatic colorings (χ-colorings). Hence, $\zeta(G) = max\{|E_\chi(G)| : over\ all\ \varphi_\chi(G)\}$.

A χ-coloring which yields $\zeta(G)$ is called a *lucky* χ*-coloring* or simply, a lucky coloring[1] and is denoted by, $\varphi_{\mathcal{L}}(G)$. The resultant graph G_ζ is called a *minimal chromatic completion graph* of G. It is trivially true that $G \subseteq G_\zeta$. Furthermore, the graph induced by the set of completion edges, $\langle E_\chi \rangle$ is a subgraph of the complement graph, $\overline{G}$. See [4] for the notion of stability in respect of chromatic completion.

A k-coloring of a graph G which yields $max\{|E_\varphi(G)| : \text{overall } k\text{-colorings}\}$ is called a lucky k-coloring.[2]

In an improper coloring an edge uv for which, $c(u) = c(v)$ is called a *bad edge*. See [5] for an introduction to defect colorings of graphs. It is observed that the number of edges of $\overline{G}$ which are omitted from E_χ is the minimum number of bad edges in a *bad chromatic completion* of a graph G.

2 Lucky 3-polynomials of paths

A path graph (or simply, a path) denoted by, P_n, is a graph on $n \geq 1$ vertices say, $V(P_n) = \{v_1, v_2, v_3, \dots, v_n\}$ and n edges namely, $E(P_n) = \{v_i v_{i+1} : i = 1, 2, 3, \dots, n-1\}$.

Recall that for λ distinct colors, $\lambda \geq \chi(G)$, the number of ways a graph G can be assigned a proper coloring is given by the chromatic polynomial of G and is denoted by, $\mathcal{P}_G(\lambda, n)$. For λ distinct colors, $\lambda \geq 3$, the path P_3 can be assigned a proper 3-coloring in $\mathcal{P}_{P_3}(\lambda, n) = \lambda(\lambda-1)(\lambda-2)$ ways. The aforesaid is equal to the number of ways a perfect lucky 3-coloring can be assigned to the path P_3 in accordance with lucky's theorem [3]. Since [3] has not been published as yet we recall lucky's theorem for perfect lucky k-coloring to be:

Theorem 1 [3] *For a positive integer* $n \geq 2$ *and* $2 \leq p \leq n$ *let integers,* $1 \leq a_1, a_2, a_3, \dots, a_{p-r}, a'_1, a'_2, a'_3, \dots, a'_r \leq n-1$ *be such that* $n = \sum\limits_{i=1}^{p-r} a_i + \sum\limits_{j=1}^{r} a'_j$ *then, the* ℓ*-completion sum-product* $\mathcal{L} = max\{\sum\limits_{i=1}^{p-r-1} \prod\limits_{k=i+1}^{p-r} a_i a_k + \sum\limits_{i=1}^{p-r} \prod\limits_{j=1}^{r} a_i a'_j + \sum\limits_{j=1}^{r-1} \prod\limits_{k=j+1}^{r} a'_j a'_k\}$ *over all possible,* $n = \sum\limits_{i=1}^{p-r} a_i + \sum\limits_{j=1}^{r} a'_j$.

[1]Note that for many graphs a lucky coloring is equivalent an equitable χ-coloring.

[2]Note that for many graphs a lucky k-coloring is equivalent an equitable k-coloring.

Note that lucky's theorem is reliant on the notion of the ℓ-completion sum-product [3]. We recall the definition to be:

Definition 2 *Let,* $t_i = \lfloor \frac{n}{\ell} \rfloor$, $i = 1, 2, 3, \ldots, (\ell - r)$ *and* $t'_j = \lceil \frac{n}{\ell} \rceil$, $j = 1, 2, 3, \ldots, r$. *Call,* $\mathcal{L} = \sum\limits_{i=1}^{\ell-r-1} \prod\limits_{k=i+1}^{\ell-r} t_i t_k + \sum\limits_{i=1}^{\ell-r} \prod\limits_{j=1}^{r} t_i t'_j + \sum\limits_{j=1}^{r-1} \prod\limits_{k=j+1}^{r} t'_j t'_k$, *the* ℓ-completion sum-product *of* n.

Also, because of the simplicity of the graph structure of paths no figure illustrations are deemed necessary for clarity. It is assumed that the reader can easily verify the vertex set partitions obtained. For path P_3 the lucky 3-polynomial is expressed as, $\mathcal{L}_{P_3}(\lambda, 3) = \lambda(\lambda - 1)(\lambda - 2)$. Note that the lucky 3-polynomial corresponds to coloring the vertex set partition, $\{\{v_1\}, \{v_2\}, \{v_3\}\}$.

Consider the path P_4. By the definition of a path a particular convention is implicit i.e. to obtain P_n from P_{n-1} we necessarily extend from v_{n-1} to v_n with the edge $v_{n-1}v_n$. Hence, it is not permissible to insert the vertex v_4 into an existing edge of P_3. The permissible lucky partitions for a lucky 3-coloring are, $\{\{v_1, v_4\}, \{v_2\}, \{v_3\}\}$, $\{\{v_1\}, \{v_2, v_4\}, \{v_3\}\}$, $\{\{v_1, v_3\}, \{v_2\}, \{v_4\}\}$. Hence, $\mathcal{L}_{P_4}(\lambda, 3) = 3\lambda(\lambda - 1)(\lambda - 2)$. Progressing to path P_5 the permissible lucky partitions for a lucky 3-coloring are found to be,

$\{\{v_1, v_4\}, \{v_2, v_5\}, \{v_3\}\}$, $\{\{v_1, v_4\}, \{v_2\}, \{v_3, v_5\}\}$, $\{\{v_1, v_5\}, \{v_2, v_4\}, \{v_3\}\}$,
$\{\{v_1\}, \{v_2, v_4\}, \{v_3, v_5\}\}$, $\{\{v_1, v_3\}, \{v_2, v_5\}, \{v_4\}\}$, $\{\{v_1, v_3\}, \{v_2, v_4\}, \{v_5\}\}$.

Hence, $\mathcal{L}_{P_5}(\lambda, 3) = 6\lambda(\lambda - 1)(\lambda - 2)$.

Progressing to path P_6 the permissible lucky partitions for a lucky 3-coloring are found to be,

$\{\{v_1, v_4\}, \{v_2, v_5\}, \{v_3, v_6\}\}$, $\{\{v_1, v_4\}, \{v_2, v_6\}, \{v_3, v_5\}\}$, $\{\{v_1, v_5\}, \{v_2, v_4\}, \{v_3, v_6\}\}$,
$\{\{v_1, v_6\}, \{v_2, v_4\}, \{v_3, v_5\}\}$, $\{\{v_1, v_3\}, \{v_2, v_5\}, \{v_4, v_6\}\}$.

Therefore, $\mathcal{L}_{P_6}(\lambda, 3) = 5\lambda(\lambda - 1)(\lambda - 2)$. Note that $\mathcal{L}_{P_6}(\lambda, 3) < \mathcal{L}_{P_5}(\lambda, 3)$.

Progressing to path P_7 the permissible lucky partitions for a lucky 3-coloring are found to be,

$\{\{v_1, v_4, v_7\}, \{v_2, v_5\}, \{v_3, v_6\}\}$, $\{\{v_1, v_4\}, \{v_2, v_5, v_7\}, \{v_3, v_6\}\}$,
$\{\{v_1, v_4, v_7\}, \{v_2, v_6\}, \{v_3, v_5\}\}$, $\{\{v_1, v_4\}, \{v_2, v_6\}, \{v_3, v_5, v_7\}\}$,
$\{\{v_1, v_5, v_7\}, \{v_2, v_4\}, \{v_3, v_6\}\}$, $\{\{v_1, v_5\}, \{v_2, v_4, v_7\}, \{v_3, v_6\}\}$,

$\{\{v_1, v_6\}, \{v_2, v_4, v_7\}, \{v_3, v_5\}\}$, $\{\{v_1, v_6\}, \{v_2, v_4\}, \{v_3, v_5, v_7\}\}$,
$\{\{v_1, v_3, v_7\}, \{v_2, v_5\}, \{v_4, v_6\}\}$, $\{\{v_1, v_3\}, \{v_2, v_5, v_7\}, \{v_4, v_6\}\}$,
$\{\{v_1, v_4, v_6\}, \{v_2, v_5\}, \{v_3, v_7\}\}$, $\{\{v_1, v_4, v_6\}, \{v_2, v_7\}, \{v_3, v_5\}\}$,
$\{\{v_1, v_5\}, \{v_2, v_4, v_6\}, \{v_3, v_7\}\}$, $\{\{v_1, v_3, v_6\}, \{v_2, v_4\}, \{v_5, v_7\}\}$,
$\{\{v_1, v_3, v_5\}, \{v_2, v_7\}, \{v_4, v_6\}\}$, $\{\{v_1, v_3, v_6\}, \{v_2, v_5\}, \{v_4, v_7\}\}$.

Therefore, $\mathcal{L}_{P_7}(\lambda, 3) = 16\lambda(\lambda-1)(\lambda-2)$.

For path P_8 the permissible lucky partitions for a lucky 3-coloring are found to be,

$\{\{v_1, v_4, v_7\}, \{v_2, v_5, v_8\}, \{v_3, v_6\}\}$, $\{\{v_1, v_4, v_7\}, \{v_2, v_5\}, \{v_3, v_6, v_8\}\}$,
$\{\{v_1, v_4, v_8\}, \{v_2, v_5, v_7\}, \{v_3, v_6\}\}$, $\{\{v_1, v_4\}, \{v_2, v_5, v_7\}, \{v_3, v_6, v_8\}\}$,
$\{\{v_1, v_4, v_7\}, \{v_2, v_6, v_8\}, \{v_3, v_5\}\}$, $\{\{v_1, v_4, v_7\}, \{v_2, v_6\}, \{v_3, v_5, v_8\}\}$,
$\{\{v_1, v_4, v_8\}, \{v_2, v_6\}, \{v_3, v_5, v_7\}\}$, $\{\{v_1, v_4\}, \{v_2, v_6, v_8\}, \{v_3, v_5, v_7\}\}$,
$\{\{v_1, v_5, v_7\}, \{v_2, v_4, v_8\}, \{v_3, v_6\}\}$, $\{\{v_1, v_5, v_7\}, \{v_2, v_4\}, \{v_3, v_6, v_8\}\}$,
$\{\{v_1, v_5, v_8\}, \{v_2, v_4, v_7\}, \{v_3, v_6\}\}$, $\{\{v_1, v_5\}, \{v_2, v_4, v_7\}, \{v_3, v_6, v_8\}\}$,
$\{\{v_1, v_6, v_8\}, \{v_2, v_4, v_7\}, \{v_3, v_5\}\}$, $\{\{v_1, v_6\}, \{v_2, v_4, v_7\}, \{v_3, v_5, v_8\}\}$,
$\{\{v_1, v_6, v_8\}, \{v_2, v_4\}, \{v_3, v_5, v_7\}\}$, $\{\{v_1, v_6\}, \{v_2, v_4, v_8\}, \{v_3, v_5, v_7\}\}$,
$\{\{v_1, v_3, v_7\}, \{v_2, v_5, v_8\}, \{v_4, v_6\}\}$, $\{\{v_1, v_3, v_7\}, \{v_2, v_5\}, \{v_4, v_6, v_8\}\}$,
$\{\{v_1, v_3, v_7\}, \{v_2, v_4, v_6\}, \{v_5, v_8\}\}$, $\{\{v_1, v_3, v_8\}, \{v_2, v_5, v_7\}, \{v_4, v_6\}\}$,
$\{\{v_1, v_3\}, \{v_2, v_5, v_7\}, \{v_4, v_6, v_8\}\}$, $\{\{v_1, v_4, v_6\}, \{v_2, v_5, v_8\}, \{v_3, v_7\}\}$,
$\{\{v_1, v_4, v_6\}, \{v_2, v_7\}, \{v_3, v_5, v_8\}\}$, $\{\{v_1, v_5, v_8\}, \{v_2, v_4, v_6\}, \{v_3, v_7\}\}$,
$\{\{v_1, v_4, v_6\}, \{v_2, v_7\}, \{v_3, v_5, v_8\}\}$, $\{\{v_1, v_3, v_6\}, \{v_2, v_4, v_8\}, \{v_5, v_7\}\}$,
$\{\{v_1, v_3, v_5\}, \{v_2, v_7\}, \{v_4, v_6, v_8\}\}$, $\{\{v_1, v_3, v_6\}, \{v_2, v_5, v_8\}, \{v_4, v_7\}\}$,
$\{\{v_1, v_4, v_6\}, \{v_2, v_5, v_7\}, \{v_3, v_8\}\}$, $\{\{v_1, v_4, v_6\}, \{v_2, v_8\}, \{v_3, v_5, v_7\}\}$,
$\{\{v_1, v_5, v_7\}, \{v_2, v_4, v_6\}, \{v_3, v_8\}\}$, $\{\{v_1, v_3, v_5\}, \{v_2, v_4, v_7\}, \{v_6, v_8\}\}$,
$\{\{v_1, v_3, v_6\}, \{v_2, v_4, v_7\}, \{v_5, v_8\}\}$, $\{\{v_1, v_4, v_6\}, \{v_2, v_8\}, \{v_3, v_5, v_7\}\}$,
$\{\{v_1, v_3, v_6\}, \{v_2, v_5, v_7\}, \{v_4, v_8\}\}$, $\{\{v_1, v_4, v_6\}, \{v_2, v_5, v_7\}, \{v_3, v_8\}\}$,
$\{\{v_1, v_5, v_7\}, \{v_2, v_4, v_6\}, \{v_3, v_8\}\}$, $\{\{v_1, v_5, v_7\}, \{v_2, v_4, v_6\}, \{v_3, v_8\}\}$,
$\{\{v_1, v_3, v_6\}, \{v_2, v_4, v_7\}, \{v_5, v_8\}\}$, $\{\{v_1, v_3, v_5\}, \{v_2, v_4, v_7\}, \{v_6, v_8\}\}$,
$\{\{v_1, v_3, v_6\}, \{v_2, v_5, v_7\}, \{v_4, v_8\}\}$.

Therefore, $\mathcal{L}_{P_8}(\lambda, 3) = 41\lambda(\lambda-1)(\lambda-2)$.

A cycle graph (or simply, a cycle) denoted by, C_n, is a graph on $n \geq 1$ vertices say, $V(C_n) = \{v_1, v_2, v_3, \dots, v_n\}$ and n edges namely, $E(C_n) = \{v_i v_{i+1} : i = 1, 2, 3, \dots, n-1\} \cup \{v_n v_1\}$. The graph structural difference between P_n and C_n is the edge $v_n v_1$. It implies that to obtain the corresponding lucky

3-polynomial, the permissible lucky partitions for $V(C_n)$ are those obtained after eliminating those lucky partitions of $V(P_n)$ with vertex subsets which have both v_1, v_n as elements. The next results follows easily without further proof.

Corollary 3 *(i)* $\mathcal{L}_{C_3}(\lambda, 3) = \lambda(\lambda-1)(\lambda-2)$,
(ii) $\mathcal{L}_{C_4}(\lambda, 3) = 2\lambda(\lambda-1)(\lambda-2)$,
(iii) $\mathcal{L}_{C_5}(\lambda, 3) = 5\lambda(\lambda-1)(\lambda-2)$,
(iv) $\mathcal{L}_{C_6}(\lambda, 3) = 4\lambda(\lambda-1)(\lambda-2)$,
(v) $\mathcal{L}_{C_7}(\lambda, 3) = 13\lambda(\lambda-1)(\lambda-2)$,
(vi) $\mathcal{L}_{C_8}(\lambda, 3) = 34\lambda(\lambda-1)(\lambda-2)$.

Recall that a null graph, $\mathfrak{N}_n$ of order n is simply an edgeless graph with vertex set, $\{v_i : 1 \leq i \leq n\}$. Constructing a path is considered to be the simplest way to add edges to a null graph to obtain a connected simple graph with minimum maximum degree, minimum number of edges and the property of symmetry. However, to find either a closed or recurrence relation between the lucky k-polynomials of null graphs and paths and cycles remains open. The table below depicts the lucky 3-polynomials for the three families of graphs for order 3 to 8.

n	$\mathfrak{N}_n$,	P_n	C_n
3	$\lambda(\lambda-1)(\lambda-2)$	$\lambda(\lambda-1)(\lambda-2)$	$\lambda(\lambda-1)(\lambda-2)$
4	$6\lambda(\lambda-1)(\lambda-2)$	$3\lambda(\lambda-1)(\lambda-2)$	$2\lambda(\lambda-1)(\lambda-2)$
5	$15\lambda(\lambda-1)(\lambda-2)$	$6\lambda(\lambda-1)(\lambda-2)$	$5\lambda(\lambda-1)(\lambda-2)$
6	$15\lambda(\lambda-1)(\lambda-2)$	$5\lambda(\lambda-1)(\lambda-2)$	$4\lambda(\lambda-1)(\lambda-2)$
7	$51\lambda(\lambda-1)(\lambda-2)$	$16\lambda(\lambda-1)(\lambda-2)$	$13\lambda(\lambda-1)(\lambda-2)$
8	$109\lambda(\lambda-1)(\lambda-2)$	$41\lambda(\lambda-1)(\lambda-2)$	$34\lambda(\lambda-1)(\lambda-2)$

Table 1.

We recall from [3] that a graph G is perfect lucky k-colorable if and only if the graph is k-colorable in accordance with lucky's theorem hence, in accordance with the lucky partition form,

$$\{\underbrace{\{\lfloor\tfrac{n}{k}\rfloor\text{-element}\}, \{\lfloor\tfrac{n}{k}\rfloor\text{-element}\}, \dots, \{\lfloor\tfrac{n}{k}\rfloor\text{-element}\}}_{(k-r)-\text{subsets}},$$
$$\underbrace{\{\lceil\tfrac{n}{k}\rceil\text{-element}\}, \{\lceil\tfrac{n}{k}\rceil\text{-element}\}, \dots, \{\lceil\tfrac{n}{k}\rceil\text{-element}\}}_{(r\geq 0)-\text{subsets}}\}.$$

First we present a lemma.

Lemma 4 *If* G *of order* n *and* $\Delta(G) \neq n-1$ *is perfect lucky* k*-colorable and* H *is a graph obtained from,* G *with one pendant vertex* v_{n+1} *added to any* $v_i \in V(G)$*, then* H *is perfect lucky* k*-colorable.*

Proof. Consider any graph G of order n and $\Delta(G) \neq n-1$ which is perfect lucky k-colorable. It implies that the G permits a proper k-coloring on the vertex set partitions of the lucky partition form,

$$\underbrace{\{\{\lfloor \tfrac{n}{k} \rfloor\text{-element}\}, \{\lfloor \tfrac{n}{k} \rfloor\text{-element}\}, \dots, \{\lfloor \tfrac{n}{k} \rfloor\text{-element}\},}_{(k-r)-\text{subsets}}$$
$$\underbrace{\{\lceil \tfrac{n}{k} \rceil\text{-element}\}, \{\lceil \tfrac{n}{k} \rceil\text{-element}\}, \dots, \{\lceil \tfrac{n}{k} \rceil\text{-element}\}\}}_{(r\geq 0)-\text{subsets}}.$$

Let graph H be, graph G with one pendant vertex v_{n+1} added to any $v_i \in V(G)$. Assume without loss of generality that in H the pendant vertex v_{n+1} is adjacent to vertex v_j.
Case 1: Assume $r > 0$. Because $\Delta(G) \neq n-1$, there exists at least one vertex partition which contains a vertex subset say, X such that, $|X| = \lceil \frac{n}{k} \rceil$ such that $v_j \in X$ and there exists at least one vertex subset say, Y such that, $|Y| = \lfloor \frac{n}{k} \rfloor$. Therefore, with regards to a lucky partition form for $V(H)$, the vertex subset $Y \cup \{v_{n+1}\}$ is permissible. It means that, the lucky partition form,

$$\underbrace{\{\{\lfloor \tfrac{n+1}{k} \rfloor\text{-element}\}, \{\lfloor \tfrac{n+1}{k} \rfloor\text{-element}\}, \dots, \{\lfloor \tfrac{n+1}{k} \rfloor\text{-element}\},}_{(k-r-1)-\text{subsets}}$$
$$\underbrace{\{\lceil \tfrac{n+1}{k} \rceil\text{-element}\}, \{\lceil \tfrac{n+1}{k} \rceil\text{-element}\}, \dots, \{\lceil \tfrac{n+1}{k} \rceil\text{-element}\}\}}_{(r+1\geq 0)-\text{subsets}},$$

yielding a vertex partition having the vertex subset $Y \cup \{v_{n+1}\}$ is a permissible to assign a perfect lucky k-coloring to graph H.
Case 2: Assume $r = 0$. By similar reasoning to that, found in Case 1 the result follows conclusively. □

Theorem 5 *Let* G *of order* $n = k(t+1) - 1$*,* $t \geq 1$ *with* $\Delta(G) \neq n-1$ *be a simple connected graph. Let* G *permit a perfect lucky* k*-coloring. Let* H *be the graph,* G *with one pendant vertex* $v_{t(k+1)}$ *added to any* $v_i \in V(G)$*. Then,* $\mathcal{L}_H(\lambda, k) < \mathcal{L}_G(\lambda, k)$*.*

Proof. Clearly the perfect lucky colorings are assigned to vertex partitions in accordance to the lucky partition form,

$$\{\underbrace{\{\lfloor\tfrac{n}{k}\rfloor\text{-element}\},\{\lfloor\tfrac{n}{k}\rfloor\text{-element}\},\ldots,\{\lfloor\tfrac{n}{k}\rfloor\text{-element}\},}_{1-\text{subset}}$$
$$\underbrace{\{\lceil\tfrac{n}{k}\rceil\text{-element}\},\{\lceil\tfrac{n}{k}\rceil\text{-element}\},\ldots,\{\lceil\tfrac{n}{k}\rceil\text{-element}\}}_{(k-1)-\text{subsets}}\}.$$

Assume without loss of generality that in H the pendant vertex v_{n+1} is adjacent to vertex v_j. Since, $\Delta(G) \neq n-1$, there exist at least two permissible vertex partitions. If we relax adjacency (allow a bad edge) then vertex v_{n+1} can only be added to all the $\{\lfloor\frac{n}{k}\rfloor$-element$\}, \{\lfloor\frac{n}{k}\rfloor$-element$\}, \ldots, \{\lfloor\frac{n}{k}\rfloor$-element$\}$ vertex subsets, over all permissible vertex partitions. For this relaxed case, $\mathcal{L}_H(\lambda, k) = \mathcal{L}_G(\lambda, k)$. Else, Lemma 2 above ensures a perfect lucky coloring and $\mathcal{L}_H(\lambda, k) < \mathcal{L}_G(\lambda, k)$. □

Theorem 3 above explains the observation that, $\mathcal{L}_{P_6}(\lambda, 3) < \mathcal{L}_{P_5}(\lambda, 3)$.

2.1 Heuristic method to determine lucky k-polynomials.

It is observed from Table 1 that $\mathcal{L}_{C_n}(\lambda, 3) < \mathcal{L}_{P_n}(\lambda, 3)$, $4 \leq n \leq 8$. The next theorem follows from this observation.

Theorem 6 *Let graph* G *be* k*-colourable and let* $H = G - e$, $e \in E(G)$. *Then,* $\mathcal{L}_H(\lambda, k) > \mathcal{L}_G(\lambda, k)$.

Proof. Because G is k-colourable it follows trivially that H is k-colourable. The lucky partitions of $V(H)$ serves as a basis to determine the permissible lucky partitions of $V(G)$ because the only graph structural difference between G and H is the edge e. Hence, with regards to G the lucky partitions of $V(H)$ which have vertex subsets which have the end-points of e as elements must be eliminated. Since, at least one such lucky partition exists, the result $\mathcal{L}_H(\lambda, k) > \mathcal{L}_G(\lambda, k)$ follows immediately. □

Let G be a graph of order n. Note that loops in G, if any, are irrelevant and may be deleted. For application of the heuristic method G is considered to be free of loops. Assume G is k-colourable.

Heuristic method:

Step 1: Since the null graph $\mathfrak{N}_n$ is k-colourable, let the set $\mathfrak{P}_0 = \{$lucky partitions of $V(\mathfrak{N}_n)\}$. Let $E(G) = \{e_i : 1 \leq i \leq \varepsilon(G)\}$. Also let $j = 0$.
Step 2: Let $i = j + 1$. Let $\mathfrak{P}_i = \mathfrak{P}_{i-1} \backslash \{$lucky partitions of $\mathfrak{P}_{i-1}$ which have vertex subsets which have the endpoints of e_i as elements$\}$.
Step 3: If $i = \varepsilon(G)$ then go to Step 4. Else, let $j = i$ and go to Step 2.

Step 4: Let $\mathcal{L}_G(\lambda, k) = |\mathfrak{P}_{\varepsilon(G)}|\lambda(\lambda-1)(\lambda-2)\cdots(\lambda-k+1)$ and exit.

Claim 2.5. The heuristic method converges and yields a unique and correct result.

Motivation. Since G is finite it implies that $\varepsilon(G)$ is finite. Hence, the iterative looping between Step 2 and Step 3 will reach go to Step 4 after $\varepsilon(G)$ iterations.
Furthermore, the lucky partitions of $V(G)$ are finite and due to the combinatorial properties of the lucky partitions all vertex subsets which have endpoints of an edge as elements are unique and finite in number. Therefore, the elimination of the corresponding lucky partitions yields a unique result. Finally, it is obvious that after exhaustive iterations, $i = 1, 2, 3, \dots, \varepsilon(G)$, the unique maximum number i.e. $|\mathfrak{P}_{\varepsilon(G)}|$, of lucky partitions remain to ensure a proper lucky k-colouring of G.

3 Conclusion

No step function or recurrence formula is known to determine $L_{P_n}(\lambda, 3)$ and $L_{C_n}(\lambda, 3)$, $n \geq 9$. Finding recurrence formula to determine lucky numbers where-after, finding a combinatorial formula to determine the number of lucky partitions which have vertex subsets without the endpoints of edges are needed to resolve these open questions.

For perfect lucky 3-colorings of paths and cycles the lucky 3-polynomial's coefficient decreases by 1 when P_{kt-1} (or C_{kt-1}), $t \geq 2$ extends to P_{kt} (or to C_{kt}). It is clear from Theorem 3 that for sufficiently large n and for $k \geq 4$ the decrease values will be greater than 1. Finding the decreases is considered a worthy avenue for research.

It is deemed worthy to have an algorithm coded to obtain the Lucky partitions of $V(\mathfrak{N}_n)$ in respect of a given lucky k-colouring. Such is needed to advance research.

References

[1] J. A. Bondy, U. S. R. Murty, *Graph Theory with Applications,* Macmillan Press, London, (1976). ⇒206

[2] F. Harary, *Graph Theory*, Addison-Wesley, Reading MA, (1969). ⇒206

[3] E. G. Mphako-Banda, J. Kok, Chromatic completion number, arXiv: 1809.01136v2. ⇒206, 207, 210

[4] E. G. Mphako-Banda, J. Kok, Stability in respect of chromatic completion of graphs, arXiv:1810.13328v1. ⇒207

[5] E. G. Mphako-Banda, An introduction to the k-defect polynomials, *Quaestiones Mathematicae*, **42,** 2 (2019) 1–10. ⇒207

[6] B. West, *Introduction to Graph Theory*, Prentice-Hall, Upper Saddle River, (1996). ⇒206

Received: October 18, 2019 • Revised: December 9, 2019

Acta Universitatis Sapientiae

The scientific journal of Sapientia Hungarian University of Transylvania (Cluj-Napoca, Romania) publishes original papers and surveys in several areas of sciences written in English.
Information about each series can be found at
http://www.acta.sapientia.ro.

Acta Universitatis Sapientiae, Informatica

Each volume contains two issues.

Sapientia University   Sciendo by De Gruyter   Scientia Publishing House

ISSN 1844-6086
http://www.acta.sapientia.ro

Information for authors

Acta Universitatis Sapientiae, Informatica publishes original papers and surveys in various fields of Computer Science. All papers are peer-reviewed.

Papers published in current and previous volumes can be found in Portable Document Format (pdf) form at the address: http://www.acta.sapientia.ro.

The submitted papers should not be considered for publication by other journals. The corresponding author is responsible for obtaining the permission of coauthors and of the authorities of institutes, if needed, for publication, the Editorial Board is disclaiming any responsibility.

Submission must be made by email (acta-inf@acta.sapientia.ro) only, using the LaTeX style and sample file at the address http://www.acta.sapientia.ro. Beside the LaTeX source a pdf format of the paper is necessary too.

Prepare your paper carefully, including keywords, ACM Computing Classification System codes (http://www.acm.org/about/class/1998) and AMS Mathematics Subject Classification codes (http://www.ams.org/msc/).

References should be listed alphabetically based on the Intructions for Authors given at the address http://www.acta.sapientia.ro.

Illustrations should be given in Encapsulated Postscript (eps) format.

One issue is offered each author free of charge. No reprints will be available.

Contact address and subscription:
Acta Universitatis Sapientiae, Informatica
Sapientia Hungarian University of Transylvania
RO 400112 Cluj-Napoca
Str. Matei Corvin nr. 4.
Email: acta-inf@acta.sapientia.ro

Printed by F&F INTERNATIONAL
Director: Enikő Ambrus

ISSN 1844-6086
http://www.acta.sapientia.ro