Department of Computer Science
Faculty of Mathematics, Physics and Informatics
Comenius University, Bratislava

Olívia Kunertová

Circular chromatic index of small snarks

(Diploma thesis)

RNDr. Ján Mazák, PhD.

Bratislava, 2017

Olívia Kunertová

Department of Computer Science
Faculty of Mathematics, Physics and Informatics
Comenius University, Bratislava

Olívia Kunertová

Circular chromatic index of small snarks

(Diploma thesis)

RNDr. Ján Mazák, PhD.

Študijný program:	Informatics
Študijný odbor:	2508 Informatics
Department:	Department of Computer Science
Advisor:	RNDr. Ján Mazák, PhD.

Bratislava, 2017

Olívia Kunertová

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Olívia Kunertová
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Circular chromatic index of small snarks
Cirkulárny chromatický index malých snarkov

Cieľ: Cieľom práce je určiť cirkulárny chromatický index malých snarkov (do 36 vrcholov) pomocou rôznych algoritmických techník, najmä inteligentného prehľadávania s návratom, prípadne celočíselného programovania. Tento cieľ je realistický (a určite zvládnuteľný minimálne pre grafy do 30 vrcholov) a získané výsledky by boli zaujímavé pre komunitu výskumníkov v oblasti farbení grafov.

Vedúci: RNDr. Ján Mazák, PhD. (od 02.12.2015)
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: prof. RNDr. Martin Škoviera, PhD.

Dátum zadania: 02.12.2015

Dátum schválenia: 16.12.2015 prof. RNDr. Rastislav Kráľovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Acknowledgements

I would like to thank my advisor Ján Mazák for his help, guiding through this work, for all answers to my questions and reading drafts of this thesis.

I want to thank to my family, because without their support I would not be able to accomplish my goals.

My thanks go to my friends who were able to cheer me up whenever I needed it. And last but not least to my friend Peťo for his support while writing thesis.

Abstrakt

Cirkulárny chromatický index grafu G dáva podrobnejšie informácie ako bežný chromatický index. Problém určenia cirkulárneho chromatického indexu grafu patrí medzi NP-úplné problémy. Cirkulárny chromatický index grafu G je najmenšie racionálne číslo r také, že graf G je r-cirkulárne hranovo ofarbiteľný. V tejto práci sa zameriavame na špeciálnu triedu grafov –snarky.

V práci navrhneme a implementujeme metódy určujúce cirkulárnu hranovú ofarbiteľnosť grafu G daným racionálny číslom r. Tieto metódy budú využité pri identifikovaní cirkulárneho chromatického indexu z množiny potenciálnych cirkulárnych indexov. Porovnáme metódy z pohľadu teoretickej časovej zložitosti ako aj experimentálne určeného času behu algoritmov. Na základe výsledkov vyberieme najlepšiu metódu, ktorá bude následne použitá pri výpočte cirkulárnych chromatických indexov malých snarkov.

Kľúčové slová: snark, cirkulárny chromatický index, cirkulárna hranová ofarbiteľnosť

Abstract

The circular chromatic index of a graph G is a refinement of a ordinary chromatic index of a graph. The problem of determining circular chromatic index of a graph belongs to class of a NP-complete problems. The circular chromatic index of a graph G is the smallest rational number r such that G is r-circular edge colorable. In this work we will focus on determining circular chromatic index of class of graphs – snarks.

We design and implemented methods determining circular edge colorability. Those methods are then used to compute circular chromatic index of a graph from given potential indices. We compare those methods based on theoretical time complexity as well as their running time. Based on the results we choose the most successful one that will be used to determine circular chromatic indices of small snarks.

Keywords: snark, circular chromatic index, circular edge colorability

Contents

Introduction

The circular chromatic index provides a more refined measure of colorability of graphs than does the ordinary chromatic index. Determining circular chromatic index of a graph is a NP-Compete problem. The circular chromatic index of a graph G, denoted as $\chi'_c(G)$ is a smallest ratio p/q of positive integers p and q for which there exists a mapping $c : E(G) \to \{0, 1, \ldots, p-1\}$ such that $q \leq |c(e) - c(f)| \leq p - q$ for edges $e, f \in E(G)$ that are incident with the same vertex.

In this work we are interested in special group of graphs – snarks that are connected bridgeless cubic graphs with chromatic index four. Unlike for 3-colorable cubic graphs for snarks the number of colors is proportional to number of edges.

In the first chapter we present brief introduction to problem of determining the circular chromatic index of a graph. We present some known results regarding computation and bounds for circular chromatic index. This bounds are later used in algorithms determining $\chi'_c(G)$ of a given graph.

Since the circular chromatic index is the smallest rational number r such that given graph G is r-circular edge colorable the second chapter provides two methods for finding the smallest number r from a given potential indices such that G is r-circular edge colorable. Also we compare those two methods based on the number of inspected potential indices as well as time complexity.

In the third chapter we introduce methods for determining whether a given graph G is r-circular colorable for a given rational number r. Then those methods can be used in algorithms presented in the second chapter for determining circular chromatic index. The algorithm description and theoretical time complexities are given. Backtrack algorithm for this problem is described. We explain how we can transform problem of determining

circular edge colorability into SAT instance. Two approaches of SAT instance versions are described.

Fourth chapter presents the algorithm that not only determines for a given rational number r and graph G whether G is r-circular edge colorable but also resolves whether $r = \chi'_c(G)$ without inspecting other potential indices. This is practical for the graphs with circular chromatic index characterized with fraction p/q such that p is small and therefore less colors are used.

The last fifth chapter is dedicated to results of implemented algorithms. We compare algorithms based on theoretical time complexities as well as experimental running time. We present circular chromatic indices that are attained by snarks of order up to 30.

Source code can be found on `https://bitbucket.org/kucierka/diploma_code`

Chapter 1

A Brief Overview of Circular Coloring

Circular coloring is a refinement of ordinary coloring. In this chapter equivalent formulations of circular coloring and basic observations of circular coloring number will be presented. Detailed information about circular colorings can be found in [15] and [16]

1.1 Circular Coloring

Let C be a circle of (euclidean) length r. We identify the circle C point by point with interval $[0, r)$ such that $0 \equiv r$. For any $x, y \in [0, r)$, the r-circular distance between x and y is described as

$$|x, y|_r = \min\{|x - y|, r - |x - y|\}.$$

In other words $|x, y|_r$ is a shortest distance of x and y on the circle C. For every $x, y \in [0, r)$, the r-circular interval is defined as follows

$$[x, y]_r = \begin{cases} [x, y] & \text{if } x \leq y, \\ [x, r) \cup [0, y] & \text{if } x > y. \end{cases}$$

Definition 1.1. *Suppose $G = (V, E)$ is a graph and C is a circle of (euclidean) length r. Let c be a mapping, assigning to each vertex x from the vertex set V an open unit length arc $c(x)$ of C. Moreover if the mapping is satisfying the condition that $c(x) \cap c(y) = \emptyset$ for every edge $e = (x, y) \in E$, we call this mapping a r-circular coloring. A graph G is*

r-circular colorable if there is an r-circular coloring of G. The circular chromatic number $\chi_c(G)$ of a graph G is defined as

$$\chi_c(G) = \inf\{r : G \text{ is } r\text{-circular colorable.}\}$$

Equivalently, the r-circular coloring of a graph G is a mapping $f : V \to [0, r)$ such that $1 \leq |f(x) - f(y)| \leq r - 1$ for every edge $(x, y) \in E$.

We can obtain an interval of length r by cutting the circle C at an arbitrary point. This interval can be identified point by point with an interval $[0, r)$. We define a mapping c' such that for each arc $c(x)$ of circle C, $c'(x)$ is the initial point of the arc $c(x)$. The arc $c(x)$ is considered going around the circle C in the clockwise direction. Therefore c' is a mapping from V to $[0, r)$ such that for every edge $(x, y) \in E : 1 \leq |c'(x) - c'(y)| \leq r - 1$. Consequently, the r-circular coloring can be presented as a mapping $c' : V \to [0, r)$ such that $1 \leq |c'(x) - c'(y)| \leq r - 1$ for every edge $(x, y) \in E$.

Similarly to the r-circular coloring, the r-interval coloring of a graph G can be defined as a mapping g, which assigns an open unit length sub-interval $g(x)$ of interval $[0, r]$ to each vertex from vertex set V such that $g(x) \cap g(y) = \emptyset$ for every edge $(x, y) \in E$. Then the chromatic number $\chi(G)$ of G is the least real number r such that there is an r-interval coloring of graph G. Also the r-interval coloring can be identified with mapping f from V to $[0, r)$. This mapping needs to satisfy following condition that for each edge $(x, y)1 \leq |f(x) - f(y)| \leq r - 1$. Furthermore for each vertex $x \in V : f(x) \leq r - 1$. So any r-interval coloring of G corresponds to an r-circular coloring of G. Also let c' be an r'-circular coloring from V to $[0, r)$. We denote $s = \max\{c'(x) : x \in V\}$, then c' is an $(s + 1)$-interval coloring of G. This leads to following result from [15]

Theorem 1.1. *For any finite graph G, $\chi(G) - 1 < \chi_c(G) \leq \chi(G)$.*

In this work we are interested in coloring of edges. Suppose $G = (V, E)$ is a graph. The line graph of the graph G is denoted as $L(G)$, which has vertex set $E(G)$. Two edges e and f are adjacent if they are incident with common vertex. The chromatic index $\chi'(G)$ of the graph G is defined as $\chi'(G) = \chi(L(G))$. Also circular chromatic index $\chi'_c(G)$ of graph G can be defined as $\chi'_c(G) = \chi_c(L(G))$. So from the Theorem 1.1 we obtain following result

$$\chi'(G) - 1 < \chi'_c(G) \leq \chi'(G).$$

1.2 Computational Methods

This section presents methods for computation of circular chromatic number. The methods were developed in the last two decades [15, 16] and are nicely summarized in the thesis [8].

Tight Cycles

Let $G = (V, E)$ be a graph with an r-circular coloring c. An edge (x, y) is called a tight edge with regard to c, if $|c(x) - c(y)|_r = 1$.

The infimum in Definition 1.1 is attained for every finite graph G and the circular chromatic numbers $\chi_c(G)$ are always rational. To prove these statements, it is useful to define a directed graph $D_c(G)$.

Definition 1.2. *Let c be a circular coloring of a graph G. Then directed graph $D_c(G)$ is defined, such that $V(D_c(G)) = V(G)$. Let x, y be two vertices. There is an edge from x to y in graph $D_c(G)$, if there is an edge $(x, y) \in E(G)$ and terminating point of interval $c(x)$ is equal to starting point of interval $c(y)$. Intervals $c(v)$ are considered as going around circle C in the clockwise direction.*

Theorem 1.2 ([15])**.** *Suppose G is finite graph and c is r-circular coloring of a graph G. If $D_c(G)$ is acyclic, then there is an r'-circular coloring c' of the graph G, such that $r' < r$ and $D_{c'}(G)$ contains a directed cycle.*

Proof. Let $D_c(G)$ be acyclic and l is mapping from vertex set V to $\mathbb{N}$. A mapping l assigns each vertex v a level, which is a length of the longest path, which ends at vertex v. Since $D_c(G)$ is acyclic, such a path exists. We denote x_0 vertex, which has a maximum level. Then the interval $c(x_0)$ can be shifted by a small distance in clockwise direction, such that disjoint intervals are assigned to adjacent vertices. We obtained r_1-circular coloring c_1 of graph G. In directed graph $D_{c_1}(G)$ vertex x_0 is isolated vertex. By repeating this process we can obtain circular coloring c'' of graph G, such that directed graph $D_{c''}(G)$ has only isolated vertices. Now we can extend each interval $c''(x)$ to a longer interval of length $s > 1$ and adjacent vertices are still assigned disjoint intervals. Now circle C can

be shrunk to a circle C' of length r/s. We obtained an r/s-circular coloring of graph G. This process can be repeated to obtain an r'-circular coloring c' of the graph G, such that $r' < r$ and $D_{c'}$ contains a directed (tight) cycle. □

Let c be an r-circular coloring of G, such that $(x_0x_1x_2 \dots x_{p-1}x_0)$ is a directed cycle in $D_c(G)$. From the definition, the union of the intervals $c(x_0), c(x_1), \dots, c(x_{p-1})$ winds around the circle C exactly q times for some integer q. Since the length of the circle C is r, the sum of the lengths of these intervals is qr. Also each interval $c(x_i)$ has a length 1, thus the sum of lengths of intervals $c(x_i)$ is equal to an integer p. Therefore the length of the circle C is $r = p/q$, for some integers p and q.

Theorem 1.3 ([15]). *If $\chi_c(G) = \frac{p}{q}$, then G has a cycle C of length kp, for some integer k. Moreover G has an independent set of size kq, which is contained in C.*

From Theorem 1.2, if G has an r-circular coloring c, G has r'-circular coloring, such that $r' \leq r$ and $r' = p/q$ for some integers p and q. Moreover p is at most the circumference (the length of a longest cycle) of G and q is at most the independence number (the size of a maximum independent set) of G.

Circumference and independence number of graph G is at most $V(G)$. There is only a finite number of rational numbers p/q, such that p and q are bounded. Thus the infimum is always attained. To determine circular chromatic number, it has to be checked, if G is r-colorable for those rational numbers. Following theorem can be obtained from theorem 1.2.

Theorem 1.4 ([15]). *If G is r-circular colorable and for every r-circular coloring c, $D_c(G)$ contains directed (tight) cycle, then $\chi_c(G) = r$. Therefore a graph G has a $\chi_c(G) = r$, if and only if G is r-circular colorable and for every r-circular coloring c of G, $D_c(G)$ contains a directed cycle.*

Although there is only a finite set of possible circular chromatic numbers for a given graph, checking each potential candidate may be a time consuming computation. This problem is NP-hard.

Suppose $r_1 < r_2 < \ldots < r_n$ are all potential values for $\chi_c(G)$ and for every r_i, there is p_i, q_i, such that $r_i = \frac{p_i}{q_i}$. If graph G is r_1-colorable, $\chi_c(G) = r_1$ and from theorem 1.4,

each r_1-coloring contains tight cycle. To find coloring, one can precolor cycle as tight cycle and try to extend this coloring. If p_1 is big, many vertices were precolored. If no r_1-coloring is found, $\chi_c(G) \geq r_2$. Now the same process can be repeated with r_2 in place of r_1. Continuing with this procedure, it can be eventually found $r_i \leq r_n$, such that there is r_i-coloring and $\chi_c(G) = r_i$. More about this method is written in section 2.

Tight Colorings

Definition 1.1 of circular chromatic number from paper [15] is equivalent to original definition of Vince, which was also mentioned in paper [15].

Definition 1.3. *For two integers $1 \leq q \leq p$, a (p, q)-coloring of a graph G is a coloring c of the vertices of G with colors $\{0, 1, \ldots, p-1\}$, such that*

$$(x, y) \in E(G) \Rightarrow p \leq |c(x) - c(y)| \leq p - q.$$

The circular chromatic number of G, denoted by $\chi_c(G)$, is defined as

$$\chi_c(G) = \inf\{k/d : \textit{there is a } (k, d)\textit{-coloring of } G.\}$$

For any integer p and $(p, 1)$-coloring c of graph G, c is an ordinary p-coloring of G.

Let c be a (p, q)-coloring and mapping $c' : V(G) \to [0, p/q)$ defined as $c'(x) = c(x)/q$. Then condition from Definition 1.3 can be formulated as

$$(x, y) \in E(G) \Rightarrow 1 \leq |c(x)/q - c(y)/q| \leq p/q - 1.$$

So (p, q)-coloring corresponds to a p/q-circular coloring of a graph G. Similarly if p/q is c'-circular coloring of a graph and mapping c defined as $c(x) = \lfloor c'(x)q \rfloor$, then c is a (p, q)-coloring of a graph G. Therefore Vince definition corresponds to Definition 1.1 and both definitions are equivalent.

If graph has an r-circular coloring c, which has no tight cycle, we will call it *loose coloring*. If a graph has an r-circular coloring c, which has tight cycle, we will call this *tight coloring*. According to the Theorem 1.2, there is an r'-coloring of graph G, such that $r' < r$. This method starts with acyclic (p, q)-coloring, where p is small, so it can be

computed exhaustively. That coloring can be improved and iteratively gain actual circular chromatic number. The following theorem describes an algorithm for improving colorings of a graph G.

Theorem 1.5 ([15]). *Let c be a (p,q)-coloring of a graph G such that every directed walk in $D_c(G)$ contains fewer than n vertices v with $p-q \leq c(v) \leq p-1$. Then G has an $(np-1, nq)$-coloring of graph G.*

Proof. For each $x \in V(G)$ let $\gamma(x)$ be the maximum number of vertices $y \neq x$ on a directed path in $D_c(G)$ ending at x with $p-q \leq c(y) \leq p-1$. Let $c'(x) = nc(x) + \gamma(x)$. c' is an $(np-1, nq)$-coloring of G.

From Definition 1.3 for each vertex $x \in V(G) : 0 \leq c(x) \leq p-1$ and from definition of $\gamma(x)$, $0 \leq \gamma(x) \leq n-1$. As a result $0 \leq c'(x) \leq pn-2$ for every vertex x.

Now an edge $(x,y) \in E(G)$ is considered. If xy is not a tight edge, then $q < |c(x) - c(y)| < p-q$. Since $|\gamma(x)-\gamma(y)| \leq n-1$, $nq \leq |c'(x)-c'(y)| \leq np-1-nq$ as required. If xy is a tight edge, $\overrightarrow{xy} \in E(D_c(G))$. Since every directed path ending at x can be extended to a path ending at y, then $\gamma(y) \geq \gamma(x)$. Now there are two cases:

1) $c(x) \leq p-q-1-c(y) = c(x)+q$ and since $0 \leq \gamma(y)-\gamma(x) \leq n-1$ and $\frac{p}{q} > 2$, then $nq \leq c'(y)-c'(x) \leq nq+n-1 \leq np-1-nq$.

2) $c(x) \geq p-q-c(y) = c(x)+q-p$ and $\gamma(y) \geq \gamma(x)+1$. Therefore $n(p-q)-(n-1) \leq c'(x)-c'(y) = n(p-q)+\gamma(x)-\gamma(y) \leq n(p-q)-1$. Now since $\frac{p}{q} > 2$, $\frac{p-1}{q} \geq 2$. Thus $n(p-q)-(n-1) \geq nq$.

So c' is an $(np-1, nq)$-coloring of G. □

Applying this algorithm to an loose (p,q)-coloring c of graph G, yields the improved coloring c' is an $(np-1, nq)$-coloring for some integer n. If c' is loose, the algorithm can be applied again to improve c'-coloring. This results in sequence c_i of (p_i, q_i)-colorings of G. Typically, after some iteration of the algorithm, p_i becomes larger than $|V(G)|$, which usually implies, that tight cycle can not exist. According to the Theorem 1.4 for none of the $p_i/q_i : \chi_c(G) = p_i/q_i$. Now let k/d be the largest valid candidate for $\chi_c(G)$ (according to Theorem 1.3). Then $\chi_c(G) \leq k/d$. Now for each $v \in V(G)$ let $\psi_i(v)$ be the rounding

of $\frac{k}{p_i}c_i(v)$ to the nearest integer. If ψ_i is a proper (k,d)-coloring of G then it is equal to $\chi_c(G)$. Otherwise algorithm from theorem 1.5 is applied and this process is repeated with better approximation.

Greedy Circular Coloring

Greedy coloring is the simplest graph coloring algorithm. It requires a permutation of the vertices of graph. Now every vertex is sequentially colored as follows. Current vertex is assigned the smallest available integer. If we have permutation $x_0, x_1, \ldots, x_{n-1}$, the algorithm colors vertex x_0 as 0 and for every $i > 0$:

$$c(x_i) = \min(\mathbb{N} \backslash \{c(x_j) : j < i \text{ and } x_j x_i \in E(G)\})$$

This greedy algorithm doesn't always yield an optimal solution, which can be seen in the figure 1.1. Although the graph is bipartite, the algorithm uses three colors.

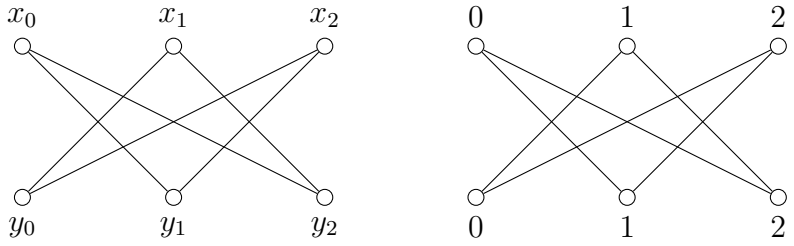

Figure 1.1: Suppose following ordering – $x_0, y_0, x_1, y_1, x_2, y_2$ for a graph on the left. Algorithm yields coloring on the right.

But there always exists permutation of vertices, for which the algorithm uses exactly $\chi(G)$ colors. Suppose any $\chi(G)$-coloring c of $G : |V(G| = n$ with the colors $0, 1, \ldots, \chi(G) - 1$, such a permutation is $x_0, x_1, \ldots, x_{n-1}$, such that:

$$c(x_0) \leq c(x_1) \leq \cdots \leq c(x_{n-1})$$

In order to compute $\chi(G)$, we need to find the permutation of vertices, for which the greedy algorithm uses minimal number of colors. This minimal number of colors is chromatic number. For this are used randomized search methods or metaheuristics.

1.3 Snarks

In this work, we are interested in snarks, that is connected bridgeless cubic graphs with chromatic index four.

By Vizing's theorem, chromatic index of cubic graphs is either three or four. In order to avoid trivial cases, snarks are restricted to be cyclically 4-edge connected with girth ≥ 5. Girth is the length of a shortest cycle. A graph is cyclically k-edge connected, if at least k-edges must be removed to disconnect it into two components, that each contains a cycle.

Table 1.1 contains numbers of snarks and general connected cubic graphs with given order. Underlined can be found in databases on websites [4] and [2]

Vertices	Snarks	Connected cubic graphs
4	0	1
6	0	2
8	0	5
10	1	19
12	0	85
14	0	509
16	0	4060
18	2	41301
20	6	510489
22	20	7319447
24	38	117940535
26	280	2094480864
28	2900	40497138011
30	28399	845480228069
32	293059	18941522184590
34	3833587	453090162062723
36	60167732	11523392072541432

Table 1.1: Table of snarks vs. general connected cubic graphs

Lower and Upper Bounds on Circular Chromatic Index of Snarks

The problem of determining circular chromatic index of a specific graph is NP-hard [15]. In order to determine the circular chromatic index of a given graph G, we need to identify bounds to all possible values that can be the circular chromatic index of G. To find an upper bound for $\chi'_c(G)$ means to find a circular edge coloring of G. To find a lower bound, it is necessary to prove that there is no such $r' < r$, such that there is r'-edge coloring.

Following Propositions concerning lower bounds were given in article [12]. We will use them to restrict number of potential indices for circular chromatic index.

Definition 1.4. *Suppose c is a (p, q)-edge coloring. A set of r consecutive colors will be called a segment of length r.*

For example the set $\{p-1, p, 1, 2\}$ is a segment of length 4. Segment colors m edges if there are exactly m edges that have colors from segment. The following propositions with proofs can be found in [12].

Proposition 1.6. *Let G be a cubic graph with $2k$ vertices, chromatic index 4 and a $(3v + u, v)$-edge-coloring. Let t and m be positive integers. If any segment of length tu colors at least m edges, then*

$$\frac{u}{v} \geq \frac{3m}{3tk - m}.$$

Then $\chi'_c(G) > 3 + m/tk$.

Next is a lower bound for snarks with girth at least five.

Proposition 1.7. *Let G be a snark on $2k$ vertices with girth at least five. Then*

$$\chi'_c(G) > 3 + \frac{2.5}{k}.$$

According to paper [6], the following proposition holds for cubic graphs.

Proposition 1.8. *There is no graph G with $\frac{11}{3} < \chi'_c(G) < 4$.*

Graph Database

Graphs in database [2] and [4] are stored in graph6 format. The following definitions are taken from McKay's website [13].

There is one object per line except optional header. All bytes are values between 63 and 126 (which are all printable ASCII characters). A file of objects is a text file.

First a representation of a bit vector needs to be described. For better comprehension Example 1.1 is given.

Example 1.1. *A bit vector x (1000101100011100) of length k can be represented as follows.*

a) Pad on the right with 0 *to make the length divisible by* 6*.*

$x = 100010110001110000$

b) Split vector into groups of 6 *bits.*

$x =$ 100010 110001 110000

c) Each group is bigendian binary number. Add 63 *to each group.*

$x =$ 97 112 111

These values are stored one per byte. Number of bytes is $\lceil \frac{k}{6} \rceil$*. $R(x)$ denotes this representation as a string of bytes of a bit vector x.*

Next we will give a representation of small nonnegative integers. The order of graph will be represented by this. Example 1.2 shows how representation of small nonnegative integer is created.

Example 1.2. *Let n be an integer in between* 0 *and* $2^{36} - 1$ (68719476735)*. There are* 3 *cases, which may arise.*

a) $0 \leq n \leq 62$ *– $N(n)$ is a single byte* $n + 63$

b) $63 \leq n \leq 258047$ *– $N(n)$ is a four bytes* 126 *$R(x)$, where x is the bigendian* 18*-bit binary form of n.*

c) $258048 \geq n \geq 68719476735$ – $N(n)$ *is a eight bytes* 126 126 $R(x)$, *where* x *is the bigendian* 36*-bit form of* n.

Examples:

a) $N(30) = 93$

b) $N(12345) = N(000011\ 000000\ 111001) = 126\ 66\ 63\ 120$

c) $N(460175067) = N(000000\ 011011\ 011011\ 011011\ 011011) = 126\ 126\ 63\ 90\ 90\ 90\ 90\ 90$

Now we can describe graph6 format of graphs. Simple undirected graphs of order between 0 and 68719476735 can be represented in graph6 format. There can be optional header (»graph6«) in a file with file extension – .g6. We give Example 1.3 for better comprehension.

Example 1.3. *Suppose* G *is of order* n. *The upper triangle of adjacency matrix of graph* G *is represented as a bit vector* x *of length* $\frac{n(n-1)}{2}$. *The following ordering of edges is used* – $(0,1),(0,2),(1,2),(0,3),(1,3),(2,3),\ldots,(n-1,n)$.

Then the graph is represented as $N(n)$ $R(x)$.

Suppose G has a 5 vertices with edges: $(0,2),(0,4),(1,3)$ and $(3,4)$. Bit vector representation of upper triangle of adjacency matrix of G is $x = 0\ 10\ 010\ 1001$.

Then $N(n)$ is 68 and $R(x) = R(010010\ 100100) = 81\ 99$. So the representation of graph G in graph6 format is $68\ 81\ 99 = DQc$.

Chapter 2

A Circular Chromatic Index

The main goal of this work is a computation of the circular chromatic index of a given snark. In this chapter we will show fundamentals of several approaches that we used for finding the circular chromatic index.

2.1 Motivation

Determining if $\chi_c(G) \leq r$ for any $r \geq 2$ is NP-Complete. As mentioned in [16] it is a consequence of the fact proved in [9] that if H is non-bipartite, then it is NP-complete to decide if an arbitrary Graph G admits a homomorphism to H. As a result a computation of the circular chromatic index $\chi'_c(G)$ of a given graph G is also NP-complete as $\chi'_c(G) = \chi_c(L(G))$.

An exhaustive search for finding circular chromatic index is not sufficient for graphs of modest size. Exhaustive search examines all possible color assignments to edges to find valid edge coloring. Therefore for (p, q)-coloring it means to check p^m edge colorings, where p is the number of colors and m is the number of edges. In our case for a snark G of order n, number of edges $m = 3n/2$. As mentioned in section 1.2 for circular chromatic number of a graph G, integers p and q are bounded by $|V(G)|$. In case of circular chromatic index, integers p and q for G are bounded by $|E(G)|$. This means that unlike for 3-colorable graphs, for snarks number of colors is proportional to its size. As consequence guesstimate is m^m edge colorings.

To prove that $\chi'_c(G) = r$, for a given graph G, we need to show that G is r-circular edge colorable and for all $r' < r$ graph G is not r'-circular edge colorable. We will describe determining index in the following section, regardless of method used for deciding whether G is r-circular edge colorable for given rational number r.

2.2 Determining circular chromatic index from potential indices

Suppose we have method for deciding whether graph G is r-circular edge colorable for given rational number r. Let S be the set of potential indices and s its size. For given graph G, we would like to decide which rational number r from given potential indices is circular chromatic index of G. In other words we need to find minimal rational number r such that G is r-circular edge colorable.

It is possible to do this by ordering potential indices by size such that we obtain sequence $r_0 < r_1 < \ldots < r_{s-1}$ of s potential indices. Now the first rational number r, from this sequence, for which is proved that G is r-circular edge colorable, is circular chromatic index. As no smaller number r' from sequence does not exists that G is r'-circular edge colorable, r is the smallest such number and the search can be terminated with result r.

This encounters problem resulting from the fact that in this ordering we try fractions that have bigger numerators first. It means that we try colorings that uses more colors. And as with increasing number of colors, computation, whether graph is circular edge colorable for a given coloring, is becoming more demanding. Thus it is desirable to avoid checking as many fractions with high numerators as possible.

Consider following ordering of potential indices

$$r_0, r_1, \ldots, r_{s-1} \qquad r_i = \frac{p_i}{q_i} \wedge \forall i, j \; i < j : p_i \leq p_j.$$

This is ordering by size of numerator. In this ordering we try potential indices with smaller numerators first, therefore colorings with smaller number of colors. To this ordering we can apply secondary sorting either by denominator or fraction. In figure 2.1 is comparison of fractions for two orderings.

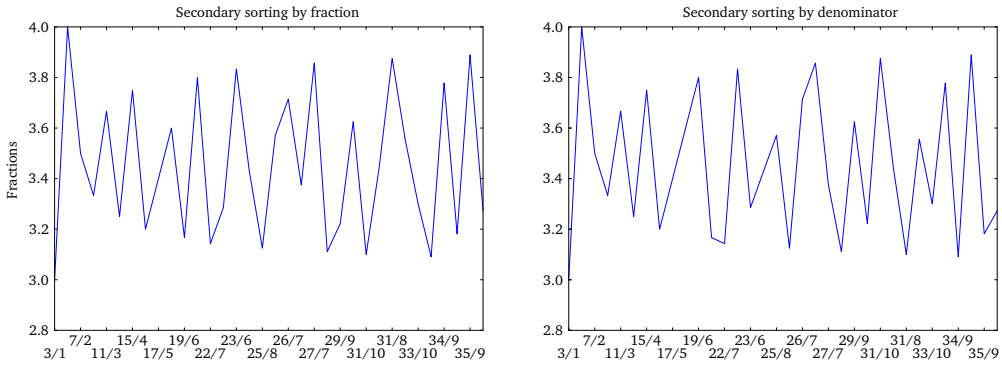

Figure 2.1: Ordering of potential indices for graph of order 24

This ordering is used for determining circular chromatic index as can be seen in algorithm 1.

Algorithm 1 Determining index

1: Set lower, upper bound to 3 and 4 respectively
2: **while** Next potential index exists **do**
3: Choose next potential index r that is in lower and upper bound
4: Determine whether G is circular edge r-colorable
5: **if** G is circular edge r-colorable **then** set upper bound to r
6: **if** G is not circular edge r-colorable **then** set lower bound to r
7: Circular chromatic index $\chi'_c(G)$ is an upper bound.

Comparison of potential indices ordering

In this subsection we will compare both orderings given in section 2.2. These orderings will be compared based on how many potential indices need to be examined whether graph G is colorable by this potential index.

First we will determine how many indices are taken to consideration for an arbitrary snark of order n with $m = 3n/2$ edges. Following expression describes number of potential

indices s_m from range $[3, 4]$ with bounded numerator and denominator by m.

$$\begin{aligned} s_m = & \sum_{i=3}^{m} \sum_{j=\lceil \frac{i}{4} \rceil}^{\lfloor \frac{i}{3} \rfloor} 1 = \sum_{i=3}^{m} (\lfloor \frac{i}{3} \rfloor - \lceil \frac{i}{4} \rceil + 1) = (m-2) + \sum_{i=3}^{m} \lfloor \frac{i}{3} \rfloor - \sum_{i=3}^{m} \lceil \frac{i}{4} \rceil = \\ = & \ (m-2) + \left(3 \frac{\left(\lfloor \frac{m}{3} \rfloor - 1 \right) \left(\lfloor \frac{m}{3} \rfloor \right)}{2} + (m \bmod 3 + 1) \lfloor \frac{m}{3} \rfloor \right) - \\ & - \left(4 \left(\frac{\lfloor \frac{m}{4} \rfloor \left(\lfloor \frac{m}{4} \rfloor + 1 \right)}{2} - 1 \right) + 2 + (m \bmod 4) \left(\lfloor \frac{m}{4} \rfloor + 1 \right) \right) \end{aligned} \tag{2.1}$$

For better intelligibility we assign values to a, b, c and d as follows.

$$\begin{aligned} a &= \lfloor \frac{m}{3} \rfloor \\ b &= m \bmod 3 + 1 \\ c &= \lfloor \frac{m}{4} \rfloor \\ d &= m \bmod 4 \end{aligned}$$

Now we can substitute these into equation 2.1.

$$\begin{aligned} s_m &= (m-2) + \left(3 \frac{(a-1)\, a}{2} + ab \right) - \left(4 \left(\frac{c\,(c+1)}{2} - 1 \right) + 2 + d\,(c+1) \right) = \\ &= (m-2) + \left(\frac{3a^2 - 3a}{2} + ab \right) - \left(4 \frac{c^2 + c - 2}{2} + 2 + cd + d \right) = \\ &= \left(\frac{2m - 4 + 3a^2 - 3a + 2ab}{2} \right) - \left(2c^2 + 2c - 4 + 2 + cd + d \right) = \\ &= \frac{1}{2} \left(2m - 4 + 3a^2 + a\,(2b-3) \right) - \left(2c^2 + c\,(d+2) - 2 + d \right) \end{aligned} \tag{2.2}$$

With back substitution we will get resulting equation for number of potential indices.

$$\begin{aligned} s_m = & \frac{1}{2} \left(2m - 4 + 3 \lfloor \frac{m}{3} \rfloor^2 + \lfloor \frac{m}{3} \rfloor \left(2\,(m \bmod 3 + 1) - 3 \right) \right) - \\ & - \left(2 \lfloor \frac{m}{4} \rfloor^2 + \lfloor \frac{m}{4} \rfloor \left((m \bmod 4) + 2 \right) - 2 + (m \bmod 4) \right) \end{aligned}$$

It can be seen that number of potential indices is in $O(m^2)$. As this number also includes fractions that are not reduced fractions, we need to look at the number of potential

indices s'_m that are represented by reduced fractions. We will show that the number of reduced fractions are also in $O(m^2)$.

$$s'_m = \left| \left\{ \frac{p}{q} \mid p, q \in \mathbb{N}\ p, q \leq m \wedge 3 \leq \frac{p}{q} \leq 4 \wedge gcd(p, q) = 1 \right\} \right|$$

In other words, we need to determine how many pairs of coprime numbers (p, q) are there such that $p, q \leq m$ and $3 \leq p/q \leq 4$.

Euler's totient function ϕ determines the number of positive integers up to given integer n that are relatively prime to the n. For example $\phi(15) = 8$. Coprime numbers to 15 that are less than 15 are $\{1, 2, 4, 7, 8, 11, 13, 14\}$. Sometimes coprime numbers to some integer n that are less than n are referred to as *totatives* of n.

$\phi(n)$ not only determines the number of coprime integers up to n but also number of positive coprime integers to n from interval $[kn + 1, (k + 1)n]$ for non negative integer k. As we can add kn to its totatives and we will get $\phi(n)$ coprime integers to n, such that they are from interval $[kn + 1, (k + 1)n]$. Therefore the number of reduced fractions from interval $[3, 4]$ with denominator q is equal to the number of totatives of q. As a result

$$1 + \sum_{i=1}^{\lceil \frac{m}{3} \rceil - 1} \phi(i) \geq s'_m \geq 1 + \sum_{i=1}^{\lfloor \frac{m}{4} \rfloor} \phi(i) \tag{2.3}$$

In order to prove that s'_m is in $\Omega(m^2)$ we will show that lower bound for s'_m from equation 2.3 is also in $\Omega(m^2)$.

$$\Phi(n) = 1 + \sum_{i=1}^{n} \phi(i) \tag{2.4}$$

The asymptotic behaviour $\Phi(n)$ can be expressed as follows [14]:

$$\Phi(n) \sim \frac{3n^2}{\pi^2} \tag{2.5}$$

From equations 2.3 and 2.5 we will get following result

$$s'_m \geq 1 + \sum_{i=1}^{\lfloor \frac{m}{4} \rfloor} \phi(i) = \Phi(\lfloor \frac{m}{4} \rfloor) \sim \frac{3\lfloor \frac{m}{4} \rfloor^2}{\pi^2} \tag{2.6}$$

Since $s_m \geq s'_m$ and both s_m and lower bound for s'_m are in $\Omega(m^2)$, the number of reduced fractions s'_m is in $\Theta(m^2)$. There are number of both all fractions and reduced fraction in the table 2.1.

n	m	# fractions	# reduced fractions
10	15	14	7
18	27	39	21
20	30	47	24
22	33	56	29
24	36	66	34
26	39	76	40
28	42	87	47
30	45	99	54

Table 2.1: Potential indices for graphs of order n with m edges

If potential indices are ordered by size in the worst case all reduced fractions need to be inspected. As the number of potential indices that are represented by reduced fractions is in $\Omega(m^2)$, the number of inspected potential indices is in $\Omega(m^2)$.

In the second case where potential indices are ordered by size of the numerator, the number of inspected potential indices is in $O(m)$. As the numerator is being increased the interval $[3, 4]$ is being split into more subintervals. This can lead to sequence of inspected fractions such that after each fraction smaller sub interval is cut off. There exists sequence of s fractions $r_0 = p_0/q_0, r_1 = p_1/q_1, \ldots, r_{s-1} = p_{s-1}/q_{s-1}$ such that

$$p_0 \leq p_1 \leq \ldots \leq p_{s-1} \ \wedge \ r_0 > r_1 > \ldots > r_{s-1} \ \wedge \ r_0 - r_1 > r_1 - r_2 > \ldots > r_{s-2} - r_{s-1}.$$

To maximize the length of this sequence we need to make consecutive fractions differences to be as small as possible. This means that as the algorithm 1 inspects each of this indices the smaller and smaller part of the interval is cut off. Suppose fraction a/b. The next fraction, that is in interval $[3, 4]$ that is smaller than a/b is $(a + 3)/(b + 1)$. Then for

consecutive fractions the difference is

$$\frac{a}{b} - \frac{a+3}{b+1} = \frac{a(b+1) - b(a+3)}{b(b+1)} = \frac{a-3b}{b(b+1)} \tag{2.7}$$

Equation 2.7 attains minimal positive value when $a = 3b+1$. As a consequence we can define i-th fraction of a sequence as $r_i = (3i+4)/(i+1)$. Figures 2.2 shows a sequence of fractions that lead to worst case of algorithm 1. i-th fraction of a sequence shrinks previous interval to its $i/(i+1)$ of previous interval length. Length of a sequence is $\lfloor m/3 \rfloor - 1$, which is in $O(m)$.

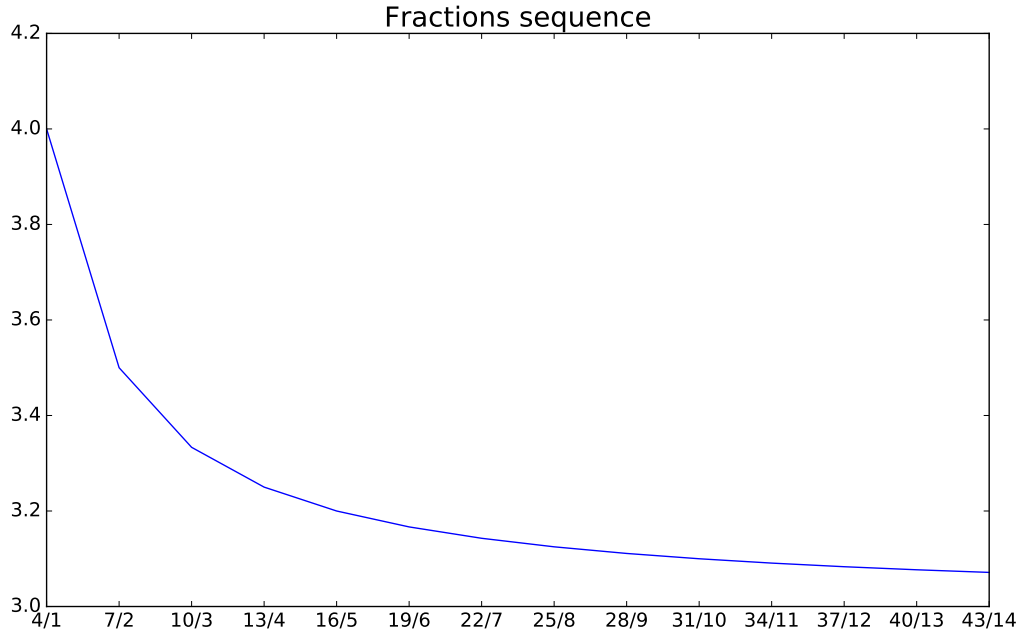

Figure 2.2: Sequence of potential indices

Based on the number of potential indices, that need to be inspected in order to determine circular chromatic index of a given graph, ordering of potential indices by numerator appears to be better than the ordering by fraction size. However in this analysis the running time of a method determining whether a given graph is circular edge colorable for a given fraction is the same for an arbitrary fraction.

We are going to analyze those orderings based on the running time of a method for

determining circular colorability of a given graph. Suppose a method that for a given graph with m edges and fraction p/q runs in a $O(p^m)$ time. We added binary search and binary search for finding circular chromatic index from potential indices with optimal medians. For a given order of graph we determine the optimal ordering of potential indices by dynamic programming. Graph describing time complexity of different algorithms can be seen in the figure 2.3. Results show that in the worst case algorithm which uses ordering of potential indices by size has the worst time complexity. Also ordinary binary search seems to be worse than the binary search with optimal medians and algorithm 1. The last two algorithms seems to have equal time complexity. Therefore we chose to use algorithm 1 to determine the circular chromatic index of a given graph.

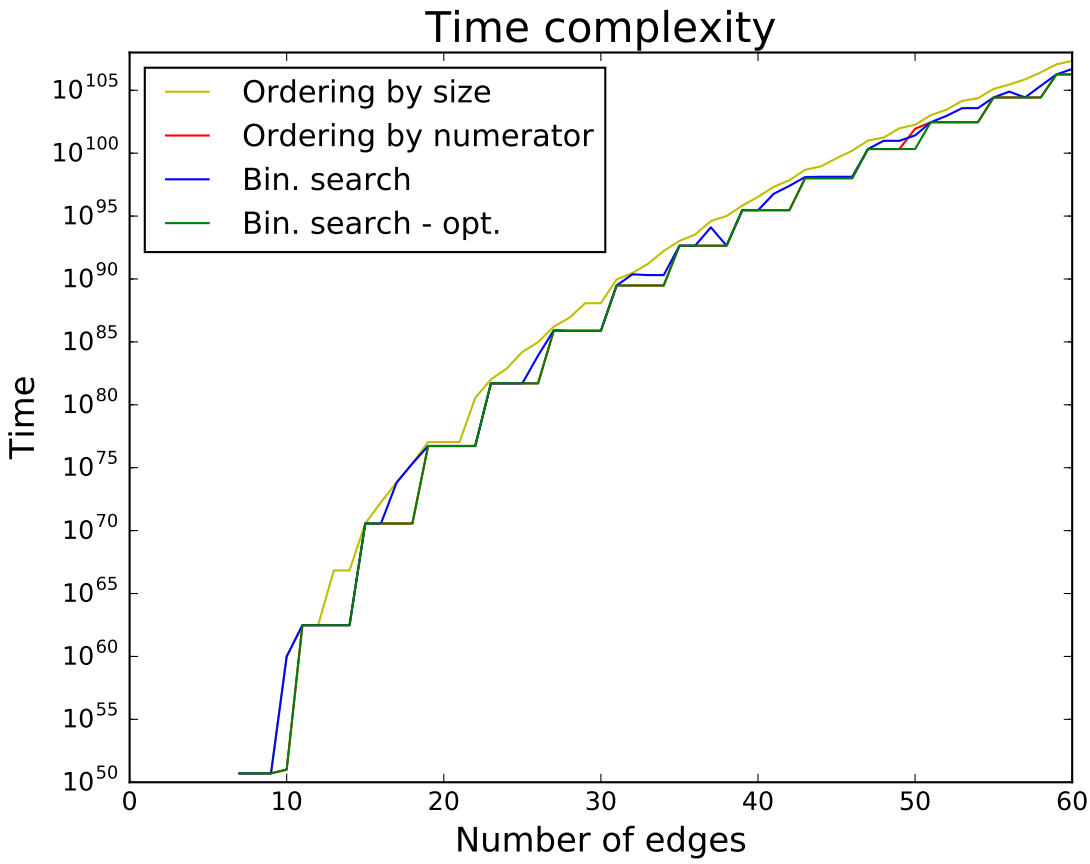

Figure 2.3: Time complexity of determining circular colorability

Chapter 3

Determining graph r-circular edge colorability

We showed that if there is a method for determining whether graph G is r-circular colorable, we can compute circular chromatic index $\chi'_c(G)$ of graph G. In this chapter we discuss few methods which decides r-circular colorability of a given graph.

3.1 Backtrack

The main idea for a backtrack algorithm is that for a particular edge the algorithm tries only colors that are not covered by its neighbours. In this section we will give detailed information about our backtrack algorithm.

Algorithm description

The input of the algorithm is a graph G and rational number r. The rational number r can be represented as a reduced fraction p/q. Algorithm uses p colors $(0, 1, \ldots, p-1)$ to color edges. Two colors c_e and c_f can be assigned to edge neighbours e and f respectively, if and only if following the condition is met

$$q \leq |c_e - c_f| \leq p - q.$$

Our algorithm is a recursive algorithm, which depth represents how many edges have

been already colored. Each edge stores an information about how many neighbours of the edge covers particular color. Feasible colors are those colors, that are covered by none of the edge's neighbours. These colors can be assigned to the edge. Before recursion we determine ordering of the edges, which they will be colored in. In each step we will choose edge that has currently the biggest number of colored neighbours.

In one step algorithm tries to assign one of the possible colors to a current edge and then moves to the next edge. If one of the colors c is assigned to a particular edge, all neighbours need to update color coverage for each color c' that does not meet the condition $q \leq |c - c'| \leq p - q$. For those colors the coverage is increased by 1.

In case of a color change of the current edge, it is necessary to cope with previous color coverages first. All neighbours of the current edge need to decrease a coverage for each color that is influenced by previous color of the current edge. Now the current edge is not colored and new color can be assigned to it.

In case the recursion is in a depth equal to the number of edges, we found a correct (p, q)-coloring and graph G is r-circular edge colorable. If the recursion never reaches such depth, it means that given graph G is not r-circular edge colorable.

Algorithm complexity

In the worst case G is not r-circular edge colorable and in order to prove that we need to inspect all possible colorings. In the previous subsection we constructed ordering of edges, that they are colored in. Now for this ordering we are going to show how many colors are tried at most for each edge.

Given graph G is connected cubic graph therefore each edge has 4 edge neighbours. For each edge the number of colored edge neighbours can be from 0 to 4 as can be seen in the figure 3.1.

As we are dealing with connected graphs only first edge has no colored edge neighbours at the time when it is chosen to be next edge in the ordering. All the other edges have at least 1 colored neighbour. Since girth of input graphs is at least 5, first three edges are incident with the same vertex.

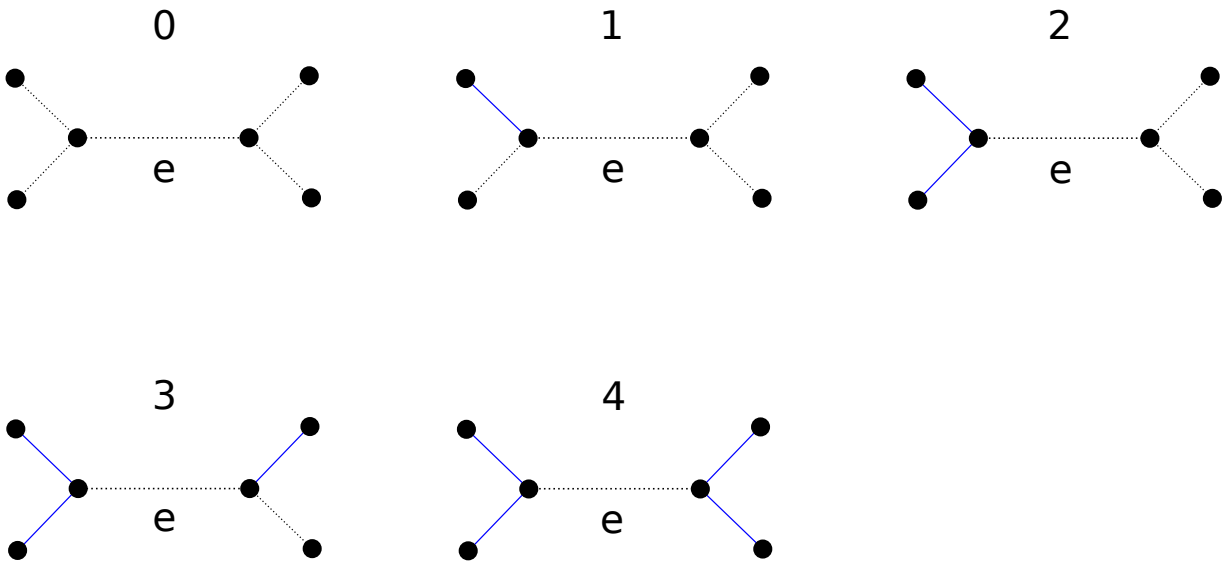

Figure 3.1: For edge e number of colored edge neighbours

- For the first edge we need to try all p colors as none of the edge neighbours is colored and therefore the number of feasible colors is equal to p.
- The second edge of ordering has one colored neighbour. As a result $2q-1$ colors are covered by this edge neighbour. Therefore the number of feasible colors for second edge is exactly $p-2q+1$ colors.
- Third edge has 2 colored edge neighbours. The first colored edge covers $2q-1$ colors. In the worst case the second edge covers only q colors that are not covered by the first edge. Therefore the number of feasible colors for the third edge is at most $p-3q+1$ colors.

Number of possible colors for an edge is proportional to number of colored neighbours. In the worst case scenario this pattern is repeated through whole ordering. A part of such ordering can be seen in the figure 3.2. We denote this ordering O.

Suppose arbitrary ordering o of edges $(e_0e_1 \dots e_{m-1})$. Let $L_o = (l_0l_1 \dots l_{m-1})$ be the sequence such that l_i is the number of colored edge neighbours of the edge e_i at the time that e_i was chosen to the ordering.

For ordering O approximate sequence $L'_O = (0, 1, 2, 1, 2, \dots, 1, 2)$. Approximate se-

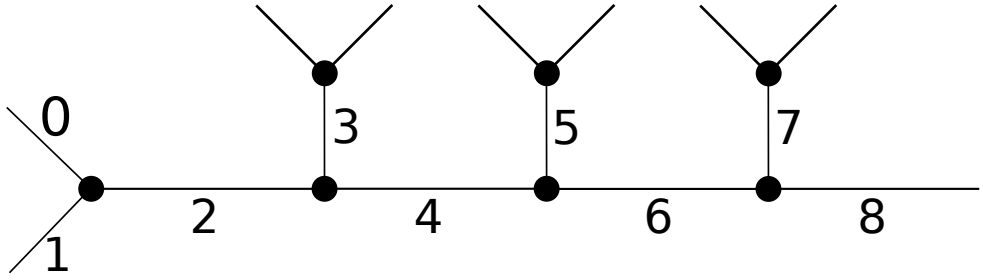

Figure 3.2: Edge ordering

quence L'_O is worse than real sequence L_O. If edge e has k colored neighbours and edge f has $k+1$ colored neighbours then e has q more possible colors than f. Let a, b, c and d be the number of 1s, 2s, 3s and 4s in sequence L. Suppose A is the sum of the number of potential colors for each edge when we consider all colors for each edge. Suppose B is the sum of number of potential colors for each edge considering ordering and colored neighbours. Then the equation 3.1 is a difference between A and B.

$$\begin{aligned} s =& a(2q-1) + b(3q-1) + c(4q-1) + d(5q-1) = \\ =& 2aq - a + 3bq - b + 4cq - c + 5dq - d = \\ =& q((a+b+c+d) + (a+2b+3c+4d)) - (a+b+c+d) = \\ =& (q-1)(a+b+c+d) + q(a+2b+3c+4d) \end{aligned} \tag{3.1}$$

The bigger the difference $A - B$ is the less colors are tried. For a given number of edges $(a+b+c+d)$ is a constant. On the other hand $(a+2b+3c+4d)$ is the sum of members of the sequence L. It can be seen from the equation 3.1 that the difference $A-B$ depends on the sum of members of the sequence L. This means that the smaller the sum of members of the sequence L the more colors are tried.

Now we need to show that sum of the L'_O is the smallest possible. Sequence L'_O contains only 1s and 2s beside one 0. Sequence L'_0 has the biggest number of 1s from all possible sequences. If ordering is constructed as described, after each 1 follows at least a number 2. Suppose there is a edge e that has 1 colored edge neighbour. Then there is a vertex v incident with edge e and vertex v has exactly 1 edge colored. After the edge e is colored, there is third edge incident with v and now it has 2 colored edge neighbours. Which

means that this edge is better candidate than any other edge that has only one colored neighbour.

As a result we have one edge that has p potential colors, $(m-1)/2$ edges that have at most $p-2q+1$ potential colors and $(m-1)/2$ edges that have at most $p-3q+1$ potential colors. Since we consider only indices from interval $[3,4]$ the following condition is met

$$p-3q+1 <= q <= \frac{p}{3} \wedge p-2q+1 \leq 2q \leq \frac{2p}{3} \qquad \frac{p}{q} \notin \{3,4\}.$$

This results in total complexity of

$$p((p-2q+1)(p-3q+1))^{\frac{m-1}{2}} \leq p\left(\frac{2p}{3}\frac{p}{3}\right)^{\frac{m-1}{2}} \in O\left(\left(\frac{\sqrt{2}}{3}\right)^{m-1} p^m\right)$$

Algorithm improvements

Theorem 3.1. *Suppose G is a graph, $r = p/q$ is rational number and $O = (e_0 e_1 \dots e_{m-1})$ is the ordering of the edges of G. Let M_c be the set of all correct r-circular edge colorings of G. Let R be a equivalence relation on M_c and $c_1, c_2 \in M_c$ then $c_1 R c_2 \Leftrightarrow c_1(e_0) = c_2(e_0)$. Let $M_{c,i}$ where $i \in [0,\dots], p-1$ be a decomposition of M_c into equivalence classes such that coloring $c \in M_{c,i} \Leftrightarrow c(e_0) = i$. Then $\forall i,j \in [0,\dots,p-1] \exists f_{i,j} : M_{c,i} \to M_{c,j}$, such that $f_{i,j}$ is a bijection.*

Proof. Suppose $M_{c,i}$ and $M_{c,j}$ are two equivalence classes. We will find bijection $f_{i,j}$. Let $k_{i,j} = (j-i) \bmod p$. Then $f_{i,j}$ is defined as follows

$$f_{i,j}(c) = c' \Leftrightarrow c'(e) = (c(e) + k_{i,j}) \bmod p.$$

To prove that $f_{i,j}$ is a bijection, we need to show that $f_{i,j}$ is both an injective and a surjective function.

Mapping $f_{i,j}$ is injective. Let $c, c' \in M_{c,i}$ then

$$\begin{aligned} f_{i,j}(c) = f_{i,j}(c') &\Rightarrow \forall e \in E(G) f_{i,j}(c)(e) = f_{i,j}(c')(e) \Rightarrow \\ &\Rightarrow \forall e \in E(G)(c(e) + k_{i,j}) \bmod p = (c'(e) + k_{i,j}) \bmod p \Rightarrow \\ &\Rightarrow \forall e \in E(G) c(e) = c'(e) => c = c' \end{aligned}$$

Mapping $f_{i,j}$ is surjective. We need to show that $\forall c' \in M_{c,j} \exists c \in M_{c,i} : f_{i,j}(c) = c'$. We will prove this by contradiction. Suppose there is coloring $c' \in M_{c,j}$ such that $\nexists c \in M_{c,i} : f_{i,j}(c) = c'$. Let c_1 be the coloring such that $\forall e \in E(G)\ c_1(e) = (c'(e) - k_{i,j}) \bmod p$. Moreover $c_1(e_0) = (c'(e_0) - k_{i,j}) \bmod p = (j - k_{i,j}) \bmod p = i$ and therefore $c_1 \in M_{c,i}$. But $f_{i,j}(c_1) = c'$ as

$$\begin{aligned}\forall e \in E(G)\ f_{i,j}(c_1)(e) =& (c_1(e) + k_{i,j}) \bmod p = \\ =& ((c'(e) - k_{i,j}) \bmod p + k_{i,j}) \bmod p = \\ =& (c'(e) - k_{i,j} + k_{i,j}) \bmod p = c'(e)\end{aligned}$$

This is a contradiction with a proposition that such a coloring does not exists. □

As a result of Theorem 3.1 in order to determine if a given Graph G is r-circular edge colorable, we need to inspect all correct colorings from only one equivalence class. This means that we can precolor one edge with one of the colors. Without loss of generality it can be the edge e_0.

We can go even further. As we showed for the second edge e_1 in the ordering 0, there is a $p - 2q + 1$ potential colors. But we need to inspect only $\lceil (p - 2q + 1)/2 \rceil$.

Similarly as in Theorem 3.1 let R_0 be a equivalence relation on $M_{c,0}$, such that for $c_1, c_2 \in M_{c,0} : c_1 R_0 c_2 \Leftrightarrow c_1(e_1) = c_1(e_1)$. Then $M_{c,0,i}$ where $i \in [q, \ldots, p - q]$ is a decomposition of $M_{c,0}$ into equivalence classes such that coloring $c \in M_{c,0,i} \Leftrightarrow c(e_1) = i$. Colors x and $(-x \bmod p)$ are the same, which means that we only need to inspect colorings from one equivalence class from pair of the classes $M_{c,0,x}$ and $M_{c,0,-x \bmod p}$.

Now as a result we have one edge that is precolored, one edge that has $\lceil (p-2q+1)/2 \rceil$ potential colors, $((m-1)/2) - 1$ edges that have at most $p - 2q + 1$ potential colors and $(m-1)/2$ edges that have at most $p-3q+1$ potential colors. This gives as total complexity

$$\begin{aligned}t =& \lceil \frac{p - 2q + 1}{2} \rceil (p - 3q + 1)((p - 2q + 1)(p - 3q + 1))^{\frac{m-1}{2}-1} \leq \\ \leq& \frac{p}{3}\ \frac{p}{3} \left(\frac{\sqrt{(2)}}{3} p \right)^{2\left(\frac{m-1}{2}-1\right)} \in O\left(\sqrt{(2)}^{m-3} \left(\frac{p}{3} \right)^{m-1} \right)\end{aligned}$$

3.2 SAT solvers

An another approach that we tried was that we transformed our problem to SAT instance and let SAT solvers to decide whether a given graph is r-circular edge colorable for given rational number r. In this section we will describe details of this process.

Satisfiability problem (SAT)

The classic satisfiability problem is to determine whether there exists boolean assignment to the variables $x_1, x_2, \ldots, x_n$, of a given boolean formula $\Phi = f(x_1, \ldots, x_n)$ such that formula evaluates to *true*.

SAT was the first problem proved to be NP-Complete. The proof, Cook's Theorem, was published in a 1971 paper [7]. The proof of SAT's NP-Completness was independently observed by Leonid Levin. Therefore the Theorem is referred to as Cook-Levin Theorem.

SAT problem instances are usually formulated in standard conjuctive normal form CNF. This means that a SAT problem is represented as conjuction of clauses. Each clause is a disjunction of literals where literal is a variable or its negation.

Examples of a SAT problem instance are

$$\Phi_1 = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1) \tag{3.2}$$

$$\Phi_2 = (x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1) \wedge (x_1 \vee x_3) \tag{3.3}$$

The formula 3.2 contains three clauses, first two consist of a disjunction of three literals and the third one is a single literal. This formula is satisfiable as there exists a assignment to the variables such that formula evaluates to *true*. Such as assignment is x_1, x_2 and x_3 are all set to *true*. By setting x_1 to *true* first and third clause is satisfied. By setting x_2 to *true* the second one is satisfied.

The formula 3.3 is unsatisfiable because there is no assignment such that Φ_2 would evaluate to *true*. Since x_1 needs to be assigned to *false* because of the third clause. This leads to the assignment of *false* and *true* to x_2 and x_3 respectively, because of the second and the forth clause. However the first clause is now unsatisfied because of this assignment.

If a restriction to number of literals in clauses are introduced, the SAT problem is

referred to as k-SAT problem. k-SAT problems means that the input formula is restricted in a way that each its clause consist of at most k literals.

DIMACS file format

If we can transform our problem into SAT instance, we can let SAT solver to decide whether G is r-circular edge colorable for a given graph G and rational number r.

Input file for the most SAT solvers is in DIMACS format described in [1].

i) The file may begin with comment lines. Each comment line starts with lower case letter c

ii) The comment lines are followed by the problem line, which begins with lower case letter p. Then problem type follows, which in our case is *cnf* and finally size of a problem, characterized by number of variables *nbvar* and number of clauses *nbclauses*.

iii) The rest of the file defines the clauses one by one

iv) Each line represent one clause

v) A clause is represented as a list of indices, positive index for positive literal and negative index for negative literal. Indices are 1-based.

vi) The representation of a clause is terminated by a final value of 0.

The formula 3.2 would be represented in DIMACS format as follows

p	cnf	3	3
1	2	−3	0
−1	2	3	0
1	0		

SAT instance for circular edge coloring

Suppose G is a graph of order n with m edges. Let c be the circular edge coloring represented by rational number r such that $r = p/q$ and p/q is a reduced fraction. The

task is to create SAT instance such that it is satisfiable if and only if G is r-circular edge colorable.

We will create SAT instance formula with mp variables and two types of clauses as follows

1. Variables – $\forall e \in E(G)$ let $P_{e,y} \Leftrightarrow c(e) = y$
2. Clause type 1 – $\forall e \in E(G) \bigvee P_{e,y} \qquad y \in \{0, 1, \ldots, p-1\}$
3. Clause type 2 – $\forall e, f \in E(G)$ (e and f are incident) – $\neg P_{e,y} \wedge P_{f,z} \qquad |y - z|_r < q$

Variable $P_{e,y}$ means that color y is assigned to edge e. Clauses of the type 1 describe that at least one color is assigned to each edge. And clauses of the type 2 describe that colors y and z satisfying the condition $|y - z|_r < q$ (in other words y and z are too close), can not be assigned to edges e and f that are incident.

Therefore if the SAT formula is satisfiable, there exists boolean assignment to the variables, such that proper coloring can be constructed.

Now we will look at the size of a SAT instance for a cubic graph G with m edges and rational number $r = p/q$. This potential coloring uses p colors, therefore we need

$$mp \text{ variables.} \tag{3.4}$$

There are m clauses of the first type as there is one for each edge.

For the number of clauses of the second type we need to determine, how many pairs of colors contradict correct circular edge coloring for two edges that are incident with the same vertex. If a one color has been already assigned to one of the edge neighbours this color eliminates $2q - 1$ colors that could be assigned to the other edge. So for each color, there are $2q - 1$ clauses for each pair of neighbour edges. However this would leave as with pair duplications. Therefore for each color we will go maximum into $q - 1$ distance from assigned color in clockwise direction. This will leave each color with q clauses. Therefore there is pq clauses for each pair of neighbour edges.

As we are dealing with cubic graphs, each edge has four neighbours. This means that there are $4m/2 = 2m$ pairs of edge neighbours. The total number of clauses is

$$nbclauses = 2mpq + m. \tag{3.5}$$

Similarly to backtrack algorithm, we can also precolor one of the edges and the second leave with half of the possible colors. We denote them as e and f. If we choose those two edges such that they are neighbours, our SAT instance will consist of

$$1 + \lceil \frac{p - 2q + 1}{2} \rceil + p(m - 2) \text{ variables.} \tag{3.6}$$

The number of clauses of the first type is the same as before but the number of clauses of the second type is different. Each edge that is incident with the edge e except edge f, results in $2q - 1$ clauses. Edge e and f do not produce any clauses, because we already restricted possible colors for edge f based on color assigned to the edge e. Each edge that is incident with the edge f except edge e result in $q\lceil (p-2q+1)/2 \rceil + (q-1)$ clauses. The pairs of edges that do not contains either edge e nor f result in pq clauses as previous SAT representation. Therefore total number of clauses for this representation is

$$nbclauses = m + 3(2q - 1) + 3\left(q\lceil \frac{p - 2q + 1}{2} \rceil + q - 1\right) + pq(2m - 7) \tag{3.7}$$

This new representation has d less clauses than previous representation, where

$$d = 7pq - 9q + 6 - 3\lceil \frac{p - 2q + 1}{2} \rceil.$$

It can be easily seen that there is at most m clauses that consist of more than two literals. Also there is at most $2mpq$ clauses that consist of exactly two literals.

Vertex SAT representation

Instead of variables for edge colors, we can try to create SAT instance where its variables represents colors in vertices. Consider colors in a node. For a 3-coloring color of a third edge is determined by the colors of first two edges. However for the $(3k + l, k)$-coloring, where $l > 0$ color of a third edge is ambiguous. As can be seen in a figure 3.3 for there are two possible colors that can be assigned to a third edge for a $(3k+1, k)$-colorings.

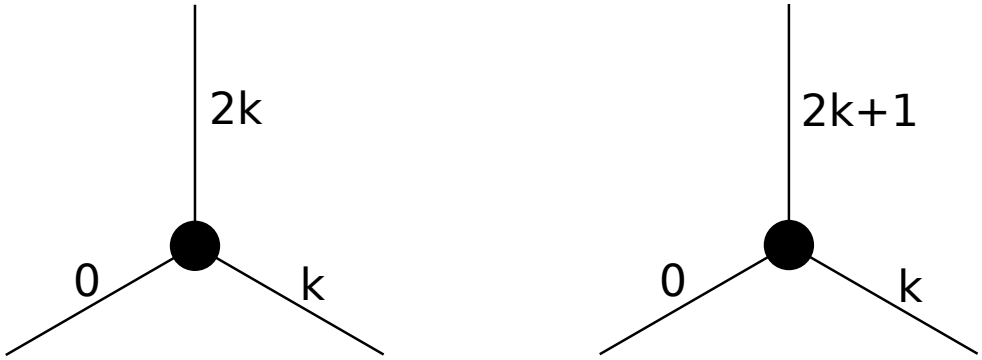

Figure 3.3: Colors in node for (3k+1, k)-coloring

Starting point 0 can be arbitrary color from $\{0, \ldots, 3k\}$. From representant triplet (a, b, c) and starting point d, we can obtain possible colors in node as follows. From representant triplet (a, b, c), we will get triplet of colors

$$((a + d) \bmod (3k + 1), (b + d) \bmod (3k + 1), (c + d) \bmod (3k + 1)).$$

In order not to duplicate triplets, we will take only 1 triplet as representant. Without loss of generality we will choose triplet $(0, k, 2k)$ as a representant. For example let $(3k+1, k)$-coloring be $10, 3$-coloring. If we chose both triplets as representants, we can get triplet of colors $(0, 3, 7)$ by two ways. The first one is that this triplet is representant, so by choosing starting point as 0 we will get this triplet. The second one is that we will take representant triplet $(0, 3, 6)$ and starting point 7. Therefore we transform representant triplet into triplet of colors $(7, 0, 3)$.

This observation can be generalized to $(3k + l, k)$-colorings, for an arbitrary $l > 0$. Representant triplets can be constructed from color differences. For example for $(3k+1, k)$-coloring, colors 0 and k have a difference 0 and colors 0 and $k + 1$ have a difference 1. Suppose $(3k + l, k)$-coloring and triplet of differences (d_0, d_1, d_2) such that $d_0 + d_1 + d_2 = l$ than color triplet constructed from differences is $(0, k + d_0, 2k + d_0 + d_1$. Moreover $3k + d_0 + d_1 + d_2 \bmod (3k + l) = 0$.

Therefore in order to find triplet representants for a $(3k + l, k)$-coloring, we need to find all unique triplet differences. Let $j = \lfloor l/3 \rfloor$, then total number of difference triplets t

is

$$\begin{aligned} t =& \sum_{i=0}^{j}(l-3i) + (l \bmod 3 \equiv 0) = l(j+1) - 3\frac{j(j+1)}{2} + (l \bmod 3 \equiv 0) = \\ =& \frac{2l(j+1) - 3j(j+1)}{2} + (l \bmod 3 \equiv 0) = \frac{(j+1)(2l-3j)}{2} + (l \bmod 3 \equiv 0) \end{aligned} \tag{3.8}$$

One triplet of colors produces multiple node colorings. Consider arbitrary edge ordering of a given vertex. In order to construct all colorings from a given triplet, we need to create all permutations of a given triplet. The number of permutations for one triplet is at most 6. As a result a total number of node colors for a given $(p,k) = (3k+l,k)$-coloring is at most $6t$. Moreover each coloring can be started with $p = 3k+l$ colors, which leads to total of *node_nbvar* variables for one node and *nbvar* variables for whole SAT instance.

$$\begin{aligned} node\_nbvar =& 6t(3k+l) = 3(j+1)(2l-3j)(3k+l) + 6(3k+l)(l \bmod 3 \equiv 0) = \\ =& 3(j+1)(2l-3j)p + 6p(l \bmod 3 \equiv 0) \geq \\ \geq& 3(j+1)lp + 6p(l \bmod 3 \equiv 0) \geq (l+1)lp + 6p(l \bmod 3 \equiv 0) = \\ =& pl^2 + pl + 6p(l \bmod 3 \equiv 0) = p(l^2 + l + 6(l \bmod 3 \equiv 0)) \end{aligned} \tag{3.9}$$

$$\begin{aligned} nbvar =& n \times node\_nbvar = \frac{2m}{3} node\_nbvar = \frac{2m}{3} p(l^2 + l + 6(l \bmod 3 \equiv 0)) = \\ =& mp\left(\frac{2}{3}\left(l^2 + l + 6(l \bmod 3 \equiv 0)\right)\right) \end{aligned} \tag{3.10}$$

Since $l < q$ total number of variables is less than $mp\left[\frac{2}{3}(q^2+q)+4\right]$.

When we compare the number of variables of SAT instance for coloring edges 3.4 versus the number of variables of SAT instance for coloring vertices 3.10, we can see that SAT instance for vertex coloring uses more variables $\forall l > 0$.

$$\begin{aligned} mp <& mp\left(\frac{2}{3}\left(l^2 + l + 6(l \bmod 3 \equiv 0)\right)\right) \\ 1 <& \frac{2}{3}\left(l^2 + l + 6(l \bmod 3 \equiv 0)\right) \end{aligned} \tag{3.11}$$

Although this alternative SAT representation uses more variables, we can still look at the number of clauses of this representation. This type of SAT instance has also two types of clauses.

1. Variables – $\forall e \in V(G)$ let $P_{v,t} \Leftrightarrow c(v) = t$
2. Clause type 1 – $\forall v \in V(G) \bigvee P_{v,t} \qquad t \in \{\text{all possible triplets of colors}\}$
3. Clause type 2 – $\forall u, v \in V(G) \wedge (u, v) \in E(G) - \neg P_{u,t_1} \wedge P_{v,t_2}$ if not he same color is assigned to the edge (u, v) by triplets t_1 and t_2.

Clauses of the first type describe that at least one of the possible triplets is assigned to the vertex. Clauses of the second type describe that if two vertices are incident with the same edge, then triplets assigned to those vertices should be compatible based on the color of this edge.

Therefore for each edge (u, v) the following equation represents the number of clauses of the second type $nbcl\_edge$.

$$\begin{aligned} nbcl\_edge =& p\left[l^2 + l + 6(l \bmod 3 \equiv 0)\right]\left[(p-1)(l^2 + l + 6(l \bmod 3 \equiv 0))\right] = \\ =& p(p-1)(l^2 + l + 6(l \bmod 3 \equiv 0)) \end{aligned} \tag{3.12}$$

Therefore total number of clauses for this alternative representation of SAT is

$$nbclauses = m(p^2 - p)(l^2 + l + 6(l \bmod 3 \equiv 0)) + m.$$

This alternative representation has more clauses than original SAT representation as the difference d between $nbclauses$ of alternative SAT instance representation and $nbclauses$ of the original SAT instance representation 3.5 is positive.

$$\begin{aligned} d =& m(p^2 - p)(l^2 + l + 6(l \bmod 3 \equiv 0)) + m - 2mpq + m = \\ =& mp^2(l^2 + l + 6(l \bmod 3 \equiv 0)) - mp(l^2 + l + 6(l \bmod 3 \equiv 0)) - 2mpq = \\ =& mp(p(l^2 + l + 6(l \bmod 3 \equiv 0)) - (l^2 + l + 6(l \bmod 3 \equiv 0)) - 2q) = \\ =& mp\left[(p-1)(l^2 + l + 6(l \bmod 3 \equiv 0)) - 2q\right] \geq mp(2p - 2 - 2q) \in \Omega(mp^2) \end{aligned} \tag{3.13}$$

This alternative representation uses both more variables and more number of clauses than original SAT representation.

Complexity of SAT solvers

According to [10] the upper bound for 3-SAT is in $O(1.32065^n)$. We can transform our general SAT instance representation into 3-SAT in order to obtain complexity.

Transforming clause into clauses containing 3 literals

In order to reduce unredtricted SAT instance into 3-SAT, we need to transform each clause $l_1 \vee l_2 \ldots \vee l_n$ for $n > 3$ into conjuction of following $n-2$ clauses

$$\begin{aligned}
&(l_1 \vee l_2 \vee x_2) \wedge \\
&(\neg x_2 \vee l_3 \vee x_3) \wedge \\
&(\neg x_3 \vee l_4 \vee x_4) \wedge \\
&\qquad \vdots \\
&(\neg x_{n-3} \vee l_{n-2} \vee x_{n-2}) \wedge \\
&(\neg x_{n-2} \vee l_{n-1} \vee l_n)
\end{aligned}$$

where $x_2 x_3 \ldots x_{n-2}$ are fresh variables that are not used anywhere else. Therefore for each unrestricted clause of length n, there are new $n-3$ variables. Although those formulas are not logically equivalent, they are equisatisfiable, which means that both formulas are satisfiable or not. However they can disagree for a particular choice of variables.

Transformed SAT instances

SAT instance formula for coloring edges introduced in section 3.2 for a given snark G of order n and rational number $r = p/q$ has following size

$$\begin{aligned}
nbvar &= 1 + \lceil \frac{p-2q+1}{2} \rceil + p(m-2) \\
nbclauses &= m + 3(2q-1) + 3\left(q\lceil \frac{p-2q+1}{2} \rceil + q - 1\right) + pq(2m-7)
\end{aligned}$$

At least $3(2q-1) + 3\left(q\lceil \frac{p-2q+1}{2} \rceil + q - 1\right) + pq(2m-7)$ clauses contains 2 literals and at most $m-1$ clauses contains p literals. Therefore in order to reduce this SAT instance into 3-SAT we need to replace $m-1$ clauses into their corresponding 3-SAT formulas. For each of this m clauses there will be $p-3$ new variables and instead of 1 clause there will

be $p-2$ clauses. Thus our new instance will have following size

$$\begin{aligned} nbvar =& \lceil \frac{p-2q+1}{2} \rceil + p(m-2) + (m-2)(p-3) + \lceil \frac{p}{2} \rceil - 2 \\ =& \lceil \frac{p-2q+1}{2} \rceil + (m-2)(2p-3) + \lceil \frac{p}{2} \rceil - 2 < \\ <& 2mp - 3m - \frac{19p}{6} + 5 = mp\left(2 - \frac{3}{p} - \frac{19}{6m} + \frac{5}{mp}\right) \\ nbclauses =& 3(2q-1) + 3\left(q\lceil \frac{p-2q+1}{2} \rceil + q - 1\right) + pq(2m-7) + 1 + \lceil \frac{p}{2} \rceil - 2 + \\ & + (m-2)(p-2) \end{aligned}$$

The second SAT instance formula that we introduced in a section 3.2 that represented coloring of vertices has for a given snark G of order n and rational number $r = p/q$ has following size

$$nbvar = mp\left(\frac{2}{3}\left(l^2 + l + 6(l \bmod 3 \equiv 0)\right)\right)$$

$$nbclauses = m(p^2 - p)(l^2 + l + 6(l \bmod 3 \equiv 0)) + m$$

There is also m unrestricted clauses as in previous SAT instance representation. Therefore there will be also $p-3$ new variables for each of the m clauses and that results in total of $m(p-3)$ new variables. Instead of 1 clause there will be $p-2$ there will be $p-2$ clauses for each one of the m unrestricted clauses. This results in new size of SAT instance representation as follows

$$\begin{aligned} nbvar =& mp\left(\frac{2}{3}\left(l^2 + l + 6(l \bmod 3 \equiv 0)\right)\right) + m(p-3) < \\ <& mp\left[\frac{2}{3}(q^2+q) + 4\right] + m(p-3) = mp\left[\frac{2}{3}(q^2+q) + 5\right] - 3m = \\ =& m\left(p\left[\frac{2}{3}(q^2+q) + 5\right] - 3\right) \\ nbclauses =& m(p^2 - p)(l^2 + l + 6(l \bmod 3 \equiv 0)) + m(p-2) \end{aligned}$$

3.3 Graph partitioning

In this work, we are focused on cyclically 4-edge connected snarks. Therefore we decided to find such 4-cuts and precolor them. Both components will be colored separately,

however they need to be color compatible on the edge cut. Many snarks are a dot product of cubic graphs G and H. Given two cubic graphs G and H a dot product $G.H$ is defined as follows.

Definition 3.1. *Choose two edges $e = ab$ and $f = cd$ in G and two adjacent vertices u and v in H. Let a', b' and v be neighbours of u and let c', d' and u be neighbours of v. Remove the edges e and f in G and vertices u and v in H. Now connect a to a', b to b', c to c' and d to d'. Result graph is called a dot product $G.H$.*

Process of creating dot product of two graphs can be seen in figure 3.4

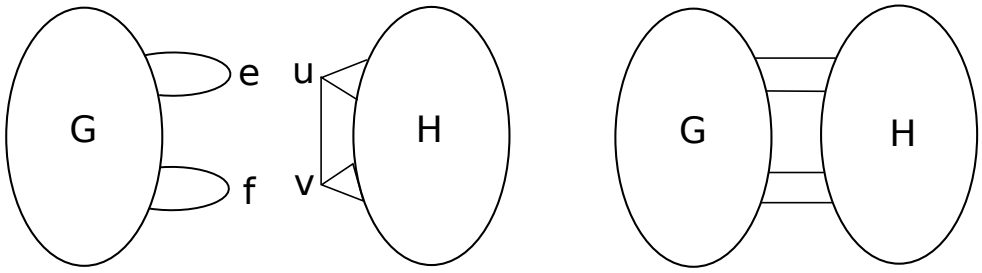

Figure 3.4: Process of creating a dot product of two cubic graphs G and H

For the given graph G and potential index $r = p/q$, method works as follows. It finds a 4-cut of a given graph, which partitions vertices of graph into 2 sets A and B and edges into 3. First edge set A_e contains edges that their both end vertices belong to set A, the second one B_e contains edges that their both end vertices belong to set B. The third edge set C_e contains cut edges.

Then 2 subgraphs are created. The first one G_A contains vertices from set A and edges both from set A_e and C_e. The second subgraph G_B contains vertices from set B and edges both from set B_e and C_e.

In each step algorithm precolors cut edges with respect to a given potential index. Those colors are assigned to respective edges both in a subgraph G_A and G_B. Subsequently it is determined if both subgraphs that has 4 edges precolored are r-circular edge colorable. If both subraphs are r-circular edge colorable in the same step of the algorithm then a graph G is also r-circular edge colorable.

Complexity

In the worst case we need to try all possible colorings for cut edges. Moreover none of the pairs of cut edges are incident with the same vertex. Therefore coloring one of the edges does not influence possible colors that can be assigned to any other cut edge. This results in p^4 possible edge colorings of a 4-cut. However similarly to backtrack improvements mentioned in section3.1, we could also precolor one of the cut edges and for the second one tried only half of the possible colorings. Therefore total number of colorings of cut edges and the maximum number of steps is

$$\frac{p^3}{2} \in O(p^3).$$

Let t be a function $G \times \mathbb{Q} \to \mathbb{N}$ that represents time complexity of determining whether a given graph is a r-circular edge colorable for a rational number r. Then time complexity of our method can be express as follows

$$\frac{p^3}{2} \left(t(G_A, r) + t(G_b, r)\right).$$

3.4 Precoloring of a tight cycle

According to Theorem 1.3 for a given potential index $r = p/q$, if r is the circular chromatic index of a given graph G then G has a cycle of a length kp for an integer k. Our next approach is based on this observation. If a $\chi'_c(G) = r$ then at least one of these cycles is tight cycle in $D_c(G)$.

Suppose G is a graph of order n with m edges and $L(G)$ is its line graph. For a given potential index $r = p/q$ we will generate all cycles of $L(G)$ of length $\{p, 2p, \ldots, kp\}$ such that $kp \leq m$. In each step algorithm precolors one of the cycles as a tight cycle. After cycle vertices are precolored we will color the rest of the vertices. If exists correct coloring the line graph $L(G)$ is r-circular colorable and therefore G is r-circular edge colorable. Otherwise we move to the next step of the algorithm. We implemented an algorithm to enumerate cycles according to paper [11].

Two kinds of methods have been introduced for enumerating all cycles in a graph. One method depends on the cycle vector space, which is formed by all cycles and edge-disjoint

union of cycles. The dimension of the cycle vector space is $\mu = e - n + 1$, where e is the number of edges and n is the number of vertices. In our case for snark of order n with $m = 3n/2$ edges line graph of order $m = 3n/2$ and $4m/2 = 3n$ edges. Therefore $\mu = 3n - 3n/2 + 1 = 3n/2 + 1$. The basis of a vector space can be obtained from spanning tree and all cycles with the ring-sum operations int he vector space. However only small part of a $2^{\mu} - 1$ vectors can be cycles. Therefore this method is very slow.

Another one is a search method. Backtracking is used to find cycles. Known upper bound for algorithm of this class is $O((n + e)(c + 1)) = O((9n/2)(c + 1))$, where c is the number of cycles. This is much faster than cycle space. Yet the process of pruning is complicated. Therefore a new way of enumerating cycles in graph was introduced in paper [11]. It can be used detect given length cycles without enumerating all cycles in the graph.

Following terms are used in algorithm description. A *path* is a alternating sequence of vertices and edges that its beginning and end are vertices, $v_0e_0v_1e_1 \dots e_{n-2}v_{n-1}$ such that every consecutive pair of vertices v_i and v_{i+1} are adjacent and incident with an edge e_i. The beginning of a path is called *head* and ending *tail*. A *simple path* is a path such that all vertices and edges except head and tail are distinct. If the head and tail are equal the simple path is called cycle. An *open path* is a simple path that is not cycle.

The main idea of the algorithm is that any k-cycle is composed of $k - 1$ length open path and an edge from tail to head of this open path. Therefore if all $k - 1$ length open path are generated, we can generate all cycles of length k ass well as all open paths of length k.

We made few changes in the algorithm described in paper, such that we do not generate all open paths of length $k - 1$ first and then from them all cycles of length k. We generate cycles sequentially, therefore we generate only one open path of a length $k - 1$ which we generate cycle of a length k from. Then we generate second open path of a length $k - 1$ and so on. We chose this approach of generating, because in case given graph is r-circular edge colorable for given r, we may not have to generate all cycles. But in a worst case scenario the last one is a tight cycle or a given graph is not r-circular edge colorable and therefore we need to generate all cycles of given length.

Our method uses cycle generating. In each step of the algorithm one cycle of a length kp for an integer k is generated such that $C = v_0e_0v_1e_1 \dots v_{kp-1}e_{kp-1}v_0$. Algorithm precolors vertices such that $c(v_i) = iq \bmod p$. Then it tries to create correct coloring for the rest of the vertices. If one is found then G is r-circular edge colorable. Otherwise algorithm moves to the next step with another generated cycle.

Chapter 4

Tight cycles

Since for the graphs that have more edges, their potential indices have bigger numerators. This means that more fractions with bigger numerators are inspected by method determining graph circular edge colorability. Therefore it would be useful to avoid this for graphs that have circular chromatic index that its numerator is small.

According to the Theorem 1.4 in order to prove that $r = \chi'_c(G)$ for a given graph G and rational number r it is not necessary to use method for determining graph circular colorability to prove that for no rational number $r' < r$ the graph G is r'-circular edge colorable. We need to prove that for each r-circular edge coloring c the given graph G contains tight cycle. Otherwise according to Theorem 1.2, there is $r' < r$ circular edge coloring of the graph G.

4.1 Generationg all circular colorings

For a given rational number r and a snark G we are going to generate all r-circular edge colorings of the graph G. We tried following methods for generating r-circular edge colorings such as

i) *Backtrack* – we alternated algorithm described in sectionbacktrack. Instead of terminating search after finding one solution, we add this solution to set of correct r-circular edge colorings.

ii) SAT solvers – if a SAT instance created from a given graph G and rational number r is satisfiable, sat solver yields a feasible solution. This solution can be transformed back to correct coloring. Therefore all colorings can be generate by following process 2.

iii) All solution SAT solvers – When SAT solver is looking for a solution, it learns a lot about the problem, but it doesn't return all that information. It just gives the solution it found. When we run the solver again, it has to re-learn all the information that was thrown away. This means that previous approach throws away useful information over and over again. Therefore using all solution SAT solvers, which give all solutions at once is more efficient.

Algorithm 2 Generate all r-circular colorings

```
1: Set variable generate_solution to True
2: Set variable result_clauses to empty vector.
3: while generate_solution do
4:     Create SAT instance for a given graph G and a potential index r
5:     for clause in result_clauses do
6:         Add negation of a clause into SAT instance
7:     Let SAT solver to decide if SAT instance formula is satisfiable
8:     if SAT instance formula is satisfiable then
9:         Add solution to result_clauses
10:    if SAT instance formula is unsatisfiable then
11:        Set generate_solution to False
12: Generate solutions from result_clauses.
```

4.2 Tight cycle check

We have generated all solutions for a given graph G and potential index r. Each one of them needs to be checked whether contains tight cycle. For each solution we create from

G a directed graph $D_c(G)$ with respect to a given coloring c as described in definition 1.2. An example of this process can be seen in figure 4.1. First graph G with coloring c such that $r = 2.5$ is transformed into a digraph that contains cycle and second graph H with coloring c' such that $r = 3$ to digraph that does not contain a cycle.

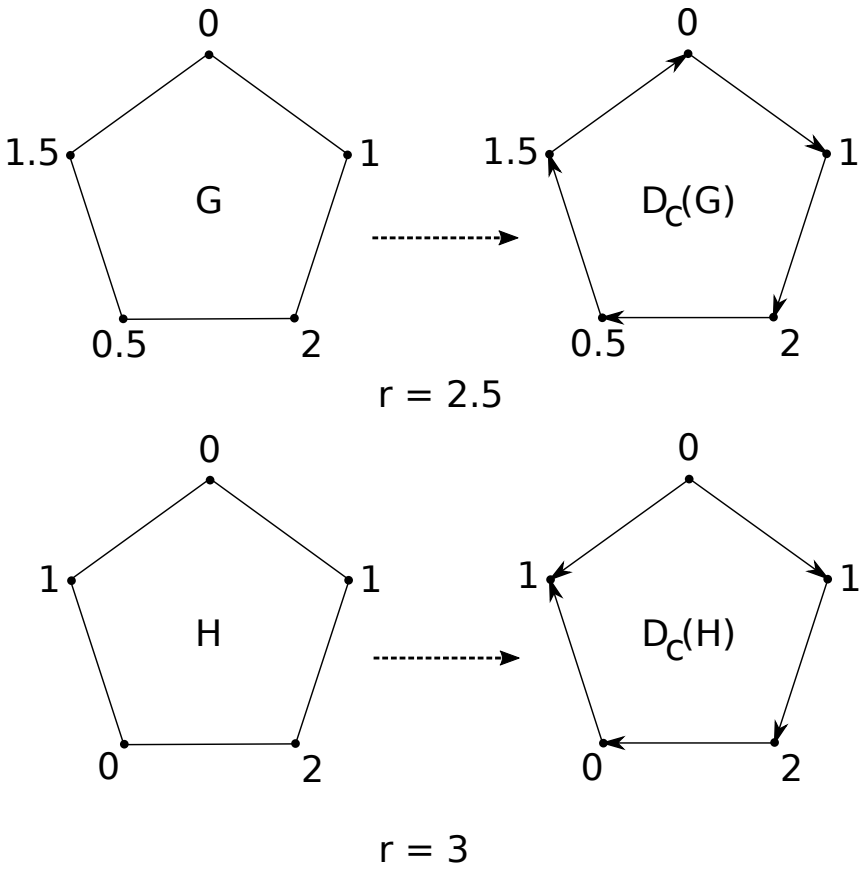

Figure 4.1: Transformation of a graph into digraph with respect to coloring c represented by rational number r

The last step is to check whether created digraph for G and coloring c contains cycle. We used depth first search to determine whether oriented graph contains cycle. Complexity of a depth first search for graph represented by adjacency list is in $O(m + n)$. Line graph of a snark contains m nodes and $2m$ edges. Therefore digraph constructed from line graph has also m nodes and at most $2m$ edges. As a result time complexity of depth first search on a directed graph created from line graph of a snark is in $O(3m) \sim O(m)$.

Chapter 5

Results

In this chapter we present results of our work. Computed circular chromatic indices of snarks will be given as well as comparison of different algorithms. We also give a comparison of theoretical and experimental time complexity.

In the table 5.1 theoretical time complexity can be seen. Backtrack algorithm is the best one of the algorithms given in the table.

Algorithm	Complexity
Exhaustive search	$O(p^m)$
Backtrack	$O\left(\sqrt{(2)}^{m-3}\left(\frac{p}{3}\right)^{m-1}\right)$
SAT instance edges	$O\left(\left(1.32065^{p\left(2-\frac{3}{p}-\frac{19}{6m}+\frac{5}{mp}\right)}\right)^m\right)$
SAT instance vertices	$O\left(\left(1.32065^{p\left(\frac{2}{3}(q^2+q)+5\right)-3}\right)^m\right)$

Table 5.1: Time complexity of method determining (p, q)-circular edge colorability of a given graph

In the following results we chose few graphs randomly and let our implemented algorithms to solve the problem. For snarks with lower order chosen graphs may repeat. For example snark of order 10 is only one, therefore we repeated computation for this graph many times. We used SAT solver [3], which won many competitions. We can see in the figure 5.1 backtrack is more efficient. However this holds only for snark of order 10. For snark of order 18 backtrack runs for several minutes although for SAT solver it takes few

seconds. The time graph for sat solver running on snarks of order 18 is in the figure 5.2 and of order 28 int the figure 5.3.

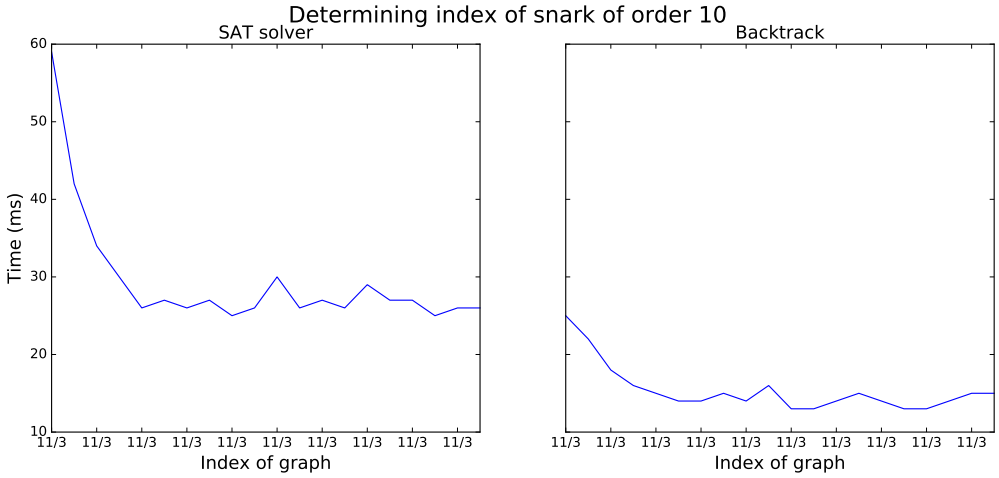

Figure 5.1: Time for snark of order 10

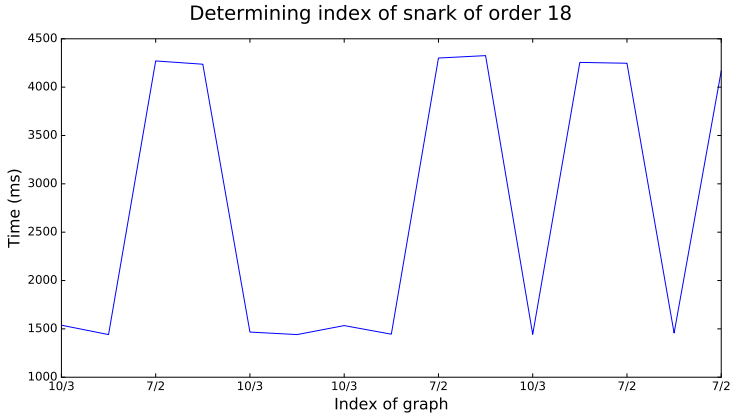

Figure 5.2: Time for snark of order 18 for a sat solver

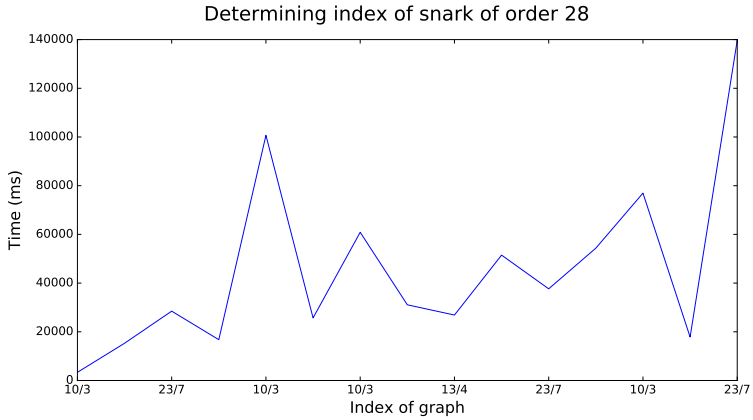

Figure 5.3: Time for snark of order 28 for a sat solver

Our backtrack algorithm has better theoretical time complexity than algorithm using SAT solver but experimental results shows that it has worse time complexity than algorithm using SAT solver.

We also compared running time of two SAT instance representations. Comparison on snark of order 10 can be seen in the figure 5.4 and of order 18 in the figure 5.5.

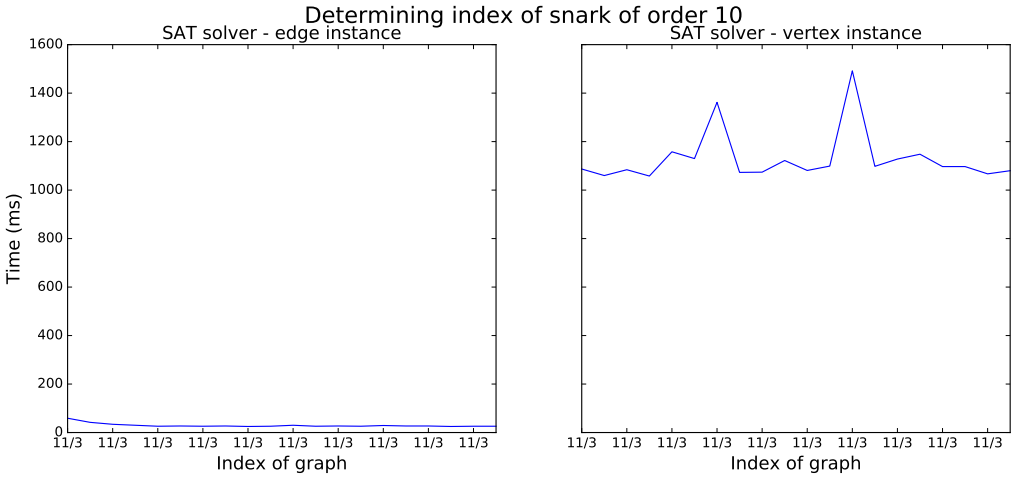

Figure 5.4: Time for snark of order 10 for two SAT instance representations

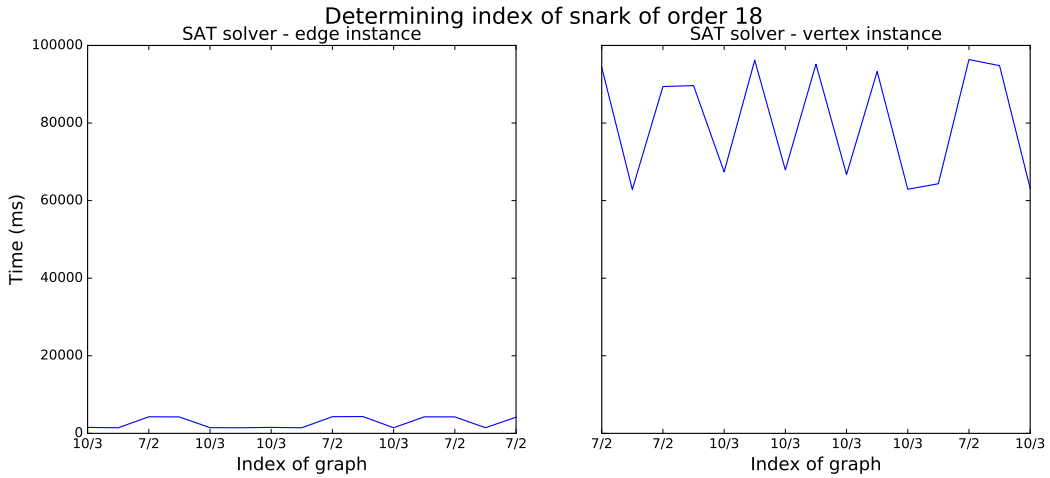

Figure 5.5: Time for snark of order 18 for two SAT instance representations

As SAT solver with instance for coloring edges has a best performance from basic algorithms. we chose this one to be the algorithm that other composed algorithms such as graph partitioning method and tight cycle method will use. And also we will use this algorithm as reference algorithm for others. The following algorithm comparison is of graph partitioning method and algorithm using SAT solvers. Figure 5.6 shows this comparison on graph of order 10.

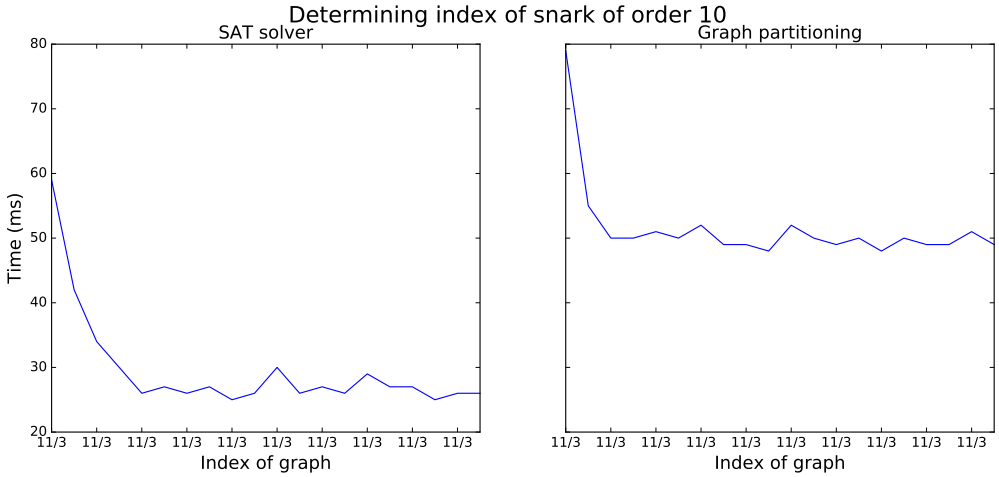

Figure 5.6: Time for snark of order 10 for two methods

Figure 5.7 shows how running time changes according to graph order. For each order

there were randomly chosen graphs. We computed average of their running times.

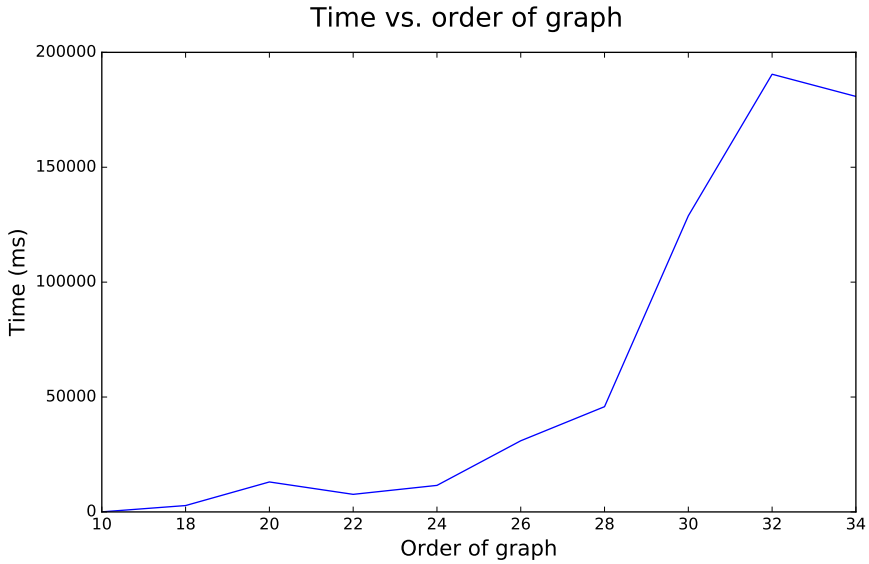

Figure 5.7: Time for snark depending on a given order

Table 5.2 shows running time of backtrack algorithm and SAT solver on snarks of order up to 30. Table 5.3 shows running time of backtrack algorithm and all solution sat with tight cycle checking on snarks of order up to 26.

Order	# snarks	Backtrack	SAT solver [3]
10	1	0m0.023s	0m0.089s
18	2		0m5.974s
20	6		0m47.081s
22	20		3m26.190s
24	38		8m27.986s
26	280		33m39.601s
28	2900		2966m27.673s
30	28399		12262m42.816s

Table 5.2: Backtrack and SAT solver comparison

Order	# snarks	$(10,3)$-index	Backtrack	All solution sat[5]
10	1	0	0m0.054s	0m0.212s
18	2	1	0m11.682s	0m7.208s
20	6	5	0m33.807s	0m11.676s
22	20	18	8m14.479s	1m48.013s
24	38	37	47m11.869s	9m47.508s
26	280	211		331m7.530s

Table 5.3: $(10,3)$-index of graphs of order less than 28

We determined indices for graphs of order up to 30 by algorithm using SAT instance representation. In the table 5.4 there are numbers of graphs with given indices.

Order		10	18	20	22	24	26	28	30
#snarks		1	2	6	20	38	280	2900	28399
Index									
(29, 9)	$3.\overline{22}$	0	0	0	0	0	**1**	0	**8**
(13, 4)	3.25	0	0	0	0	0	**13**	**314**	**4130**
(23, 7)	3.29	0	0	0	0	**1**	**55**	**1076**	**12775**
(33, 10)	3.30	0	0	0	0	0	0	0	**1**
(10, 3)	$3.\overline{33}$	0	**1**	**5**	**18**	**37**	**211**	**1509**	**11483**
(17, 5)	3.40	0	0	**1**	**2**	0	0	**1**	**2**
(7, 2)	3.50	0	**1**	0	0	0	0	0	0
(11, 3)	$3.\overline{66}$	**1**	0	0	0	0	0	0	0

Table 5.4: Results for graphs of order less than or equal to 30

Conclusion

In this work we dealt with the problem of determining circular chromatic index. The circular chromatic index provides a more refined measure of colorability of graphs than does the ordinary chromatic index. This problem belongs to class of NP-Compete problems. In order to determine the circular chromatic index $\chi'_c(G)$ of a given graph G it is necessary to find the smallest fraction $r = p/q$ of positive integers p and q for which G is r-circular edge colorable.

We focused on special class of graphs – snarks, connected bridgeless cubic graphs with chromatic index four.

We presented methods for finding smallest rational number r such that given graph G is r-circular edge colorable providing we are able to decide if G is circular edge colorable for a given potential index. We designed several methods for determining circular edge colorability of graphs. Those methods were compared based on theoretical time complexity as well as experimental running time.

We also implemented method that not only determines if a given graph G is circular edge colorable for a given potential index but also decides if this potential index is circular chromatic index of G without inspecting other potential indices.

The results of methods are also presented. Based on the results the most successful method in practise was one that transforms problem of r-circular edge colorability of a given graph G and potential index r into SAT instance and than SAT solver is used to decide satisfiability of this formula.

All circular chromatic indices of snarks of order up to 30 that are presented in the results chapter were computed by this method.

Bibliography

[1] Cnf files. http://people.sc.fsu.edu/~jburkardt/data/cnf/cnf.html. (Accessed on 04/23/2017).

[2] House of graphs - snarks. http://hog.grinvin.org/Snarks. (Visited on 01/02/2016).

[3] Lingeling, plingeling and treengeling. http://fmv.jku.at/lingeling/. (Accessed on 03/19/2017).

[4] Gordon royle's cubic graphs. http://staffhome.ecm.uwa.edu.au/~00013890/remote/cubics/#snarks, 1996. (Visited on 01/02/2016).

[5] Cnf2obdd or bdd minisat all. http://www.sd.is.uec.ac.jp/toda/code/cnf2obdd.html, 2016. (Accessed on 03/19/2017).

[6] Peyman Afshani, Mahsa Ghandehari, Mahya Ghandehari, Hamed Hatami, Ruzbeh Tusserkani, and Xuding Zhu. Circular chromatic index of graphs of maximum degree 3. *Journal of Graph Theory*, 49(4):325–335, 2005.

[7] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM.

[8] Mohammad Ghebleh. Theorems and Computations in Circular Colourings of Graphs, 2007.

[9] Pavol Hell and Jaroslav Nešetřil. On the complexity of h-coloring. *Journal of Combinatorial Theory, Series B*, 48(1):92 – 110, 1990.

[10] Timon Hertli, Robin A. Moser, and Dominik Scheder. Improving PPSZ for 3-sat using crtitical variables. *CoRR*, abs/1009.4830, 2010.

[11] Hongbo Liu and Jiaxin Wang. A new way to enumerate cycles in graph. In *Proceedings of the Advanced Int'L Conference on Telecommunications and Int'L Conference on Internet and Web Applications and Services*, AICT-ICIW '06, pages 57–, Washington, DC, USA, 2006. IEEE Computer Society.

[12] Martin Macaj and Ján Mazák. Asymptotic lower bounds on circular chromatic index of snarks. *Electr. J. Comb.*, 20(2):P2, 2013.

[13] Brendan McKay. users.cecs.anu.edu.au/~bdm/data/formats.txt. `http://users.cecs.anu.edu.au/~bdm/data/formats.txt`. (Visited on 01/03/2016).

[14] Ilan Vardi. *Computational Recreations in Mathematica*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1991.

[15] Xuding Zhu. Circular chromatic number: a survey. *Discrete Mathematics*, 229(1-3):371–410, 2001.

[16] Xuding Zhu. Recent developments in circular colouring of graphs. *Algorithms and Combinatorics*, 26:497–550, 2006.