DIMACS-DIMATIA International REU
Research Experience for Undergraduates 2014

June 1 – August 1 2014,

Piscataway, NJ, USA and Prague, Czech Republic

Dušan Knop (ed.)

DIMACS/DIMATIA International REU 2014

Published by IUUK–ITI series 2014-619.
Centre of Excellence – Institute for Theoretical Computer Science,
Faculty of Mathematics and Physics, Charles University
Malostranské náměstí 25, 118 00, Praha 1, Czech Republic

Published by MATFYZPRESS,
Publishing House of the Faculty of Mathematics and Physics,
Charles University in Prague,
Sokolovská 83, 186 75 Prague 8, Czech Republic
as the 478th publication

Prague 2013

ISBN 978-80-7378-284-9

Preface

Charles University in Prague and particularly Department of Applied Mathematics (KAM), Computer Science Institute of Charles University (IUUK) and its international centre DIMATIA, are very proud that they are hosting one of the very few International REU programmes which are funded jointly by NSF and the Ministry of Education of Czech Republic (under the framework of Kontakt programmes ME 521, ME 886 and ME 09074). This programme is a star programme at both ends and it exists for more than a decade since 2001. Repeatedly, it has been awarded for its accomplishments and educational excellence.

The program of Kontakt on the Czech side was not renewed for year 2014 and thus the programme was financed jointly by Section Informatics of MFF and our grants CE-ITI P202/12/G061, ERCCZ LL1201 and SVV—202-09/260103 (Discrete Models and Algorithms). We thank all the contributors and hope that the next year will bring us a stable support.

This booklet reports just the programme in 2014. I thank to Dušan Knop, the Czech mentor of this year, for a very good work both during the programme itself and after.

Prague, October 17, 2014

Jaroslav Nešetřil

The Center for Discrete Mathematics and Theoretical Computer Science (DIMACS) at Rutgers University, New Jersey and the Center for Discrete Mathematics, Theoretical Computer Science and Applications (DIMATIA) have collaborated on a joint Research Experiences for Undergraduates (REU) program for more than a decade. The program entails nine weeks of intensive research during the summer. It is highly competitive with approximately 25 American students being selected from more than 300 applications. The first part of the program is held at the DIMACS offices at Rutgers University with both Czech and American students working on open problems with their mentors. The American side of the program consists of regular lunches and teas, a weekly seminar series hosting both local speakers and renowned outside speakers and workshops devoted to such topics as graduate school applications, ethical behavior in research, and field trips to DIMACS industrial partners. Weekly Bridge Workshops are held in which Czech and American students are given the opportunity to work together on problems in discrete mathematics. Four American students are selected to return to Prague with the Czech students to participate in the second part of the program at DIMATIA. At DIMATIA, the students continue to work on their research projects and participate in daily seminars and lectures as well as attend the Midsummer Combinatorial Workshop.

The DIMACS/DIMATIA REU joint collaboration provides students with a unique cultural experience providing students valuable insight into research activities on an international scale.

Eugene Fiorini
Associate Director, DIMACS

DIMACS/DIMATIA Research Experiences for Undergraduates (REU) is a joint program of the DIMATIA center, Charles University in Prague and DIMACS center, Rutgers University, New Jersey. This year's participants from Charles University were students Pavel Dvořák, Lukáš Folwarczný, Peter Korcsok, Karel Král and Veronika Steffanová. Their coordinator was Dušan Knop, who participated in the scientific work, but mainly took care of organizing the DIMATIA part of the program. Together with more than thirty students from universities from all over the United States, they participated in the first part of the program, at Rutgers University of New Jersey in Piscataway, USA, from June 1st to July 19thst. Four American students were selected to join, together with their coordinator, the Czech students in the second part which took place at Charles University in Prague from July 21st to August 1st. The students were David DeSimone, Dylan Quintana, Zachary Wheeker and Elizabeth Yang. The coordinator was Rachel Levanger.

The first part of the program mainly consists of students solving open mathematical problems brought by their mentors. Students attended several lectures and they also participated in a trip to IBM which was organized by DIMACS. Here the students heard about applications of mathematics and computer science.

In Prague, the students attended a series of lectures given by professors mainly from the Department of Applied Mathematics and the Computer Science Institute of Charles University. They also had the opportunity to attend the Midsummer Combinatorial Workshop.

In addition to the scientific program, an important part of the REU is an intercultural experience. During the first part, whole day was dedicated to presentations of Czech Republic and cultures from which the American students come from. The students participated together in informal sport activities and sightseeing trips.

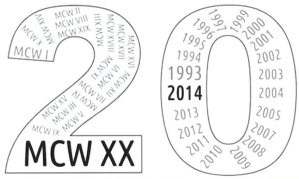
The students got important experiences with research and life abroad. For some of them, the program will certainly be an important milestone in their future scientific career.

This booklet presents the results of the Czech students stemming from the REU programme and reports of the American students about their visit to Prague. I would like to thank Josef Cibulka and Martin Balko for providing the source files for this booklet.

Dušan Knop

The participants of the Internation REU programme at Rutgers University.

The participants of the Prague part of the programme.



Midsummer Combinatorial Workshop

Prague 28.7. - 1.8. 2014

Midsummer Combinatorial Workshop—conference group photo.

Contents

Lecture Notes on Memory Efficiency in Modeled RAM/Disk Systems (lecture by Michal Koucký)

David DeSimone

We may have the situation where we have a program, and that program runs out of memory. We have several solutions to this problem. One solution would be to swap important information into the Hard Disk, treating it almost like RAM. However the situation may be that the Hard Disk is also full, and swapping may not be an option. In this case we would like a more creative solution that can use the Hard Disk for memory storage without disturbing any existing data present.

We start by examining two situations, the situation that our computer system has only RAM, and the situation where our computer system has a full Hard Disk. It will be shown that, for certain computations, it is preferable to have a full hard disk over no hard disk at all. Let us model the second situation in terms of a classical Turning Machine. In this case, our TM has one read-only input tape, multiple read/write tapes referred to as the "Working Memory", one write-only output tape, and one read/write "Hard Drive" tape.

The first problem we will examine in this situation is that of basic matrix multiplication. What is the memory needed to compute $A \cdot B = C$, asssuming that A and B are nxn matrices. One can see that using a trivial matrix multiplication approach, one would only need O(log(n))-bits of memory for the multiplication. This is due to that fact that we only need to store one non-output number at any time, $\sum_k A_{ik}B_{kj}$, which corresponds to the particular resultant matrix entry we are calculating.

Now let's not just consider two matrices, but the general case of n matrices We could follow a naive approach, and directly calculate each entry in the resultant matrix through the summation formula. This would results in something like O(nlog(n)). However this problem can be done far more efficiently. It is possible to do this computation in $O(log^2(n))$ time.

As an aside, this problem brings us into the issue of NLog Space-completeness. The problem of n-matrix multiplication in known as NLog Space-complete. An important problem in complexity theory is if LOG = NLOG. In order to solve the n-matrix multiplication problem, we will examine another problem that may seem unrelated at first. However this solution to this problem will help us greatly in developing the solution to n-matrix multiplication. The problem is the following:

We can express arithmetic expressions in terms in binary trees. Given the expression (15 * 20) + 35, we imagine it's binary tree representation, where each number is a leaf of the tree, and each operator has exactly two children joining the two things it operates on. Given such a tree, what is the minimum amount of storage needed to calculate the end result of the expression?

We could preform a modified version of DFS on this tree to calculate the result, however we would need d total "numbers" stored, where d is the depth of the tree. Note that we will restrict our "numbers" to be finite, or else we could just cheat and play games concatenating numbers together like strings for efficient storage.

However, it is possible to solve this problem only storing **3** numbers. A caveat to this, is that whatever we are acting on must form a Ring, (R, +, *). For this lecture we say that a ring is an algebraic structure such that the set forms a group under the addition, but need not form a group under the multiplication. For this problem, we would like to shift of model slightly to work in terms of general registers. These general registers have limited operations. In fact, they only have two operations.

For register r_i,
$r_i \leftarrow r_i + r_j[\cdot r_k]\, where\, i \neq j,\, k$
$r_i \leftarrow r_i + r_j[\cdot constant]\, where\, i \neq j.$

Let us assume initially that all registers contain 0, except r_2, which contains 1. We define our goal to be to calculate a "formula" $f(x_1, x_2, \ldots x_n)$ that captures the value of the expression until a particular point of the statement. We will do this be defining an inductive argument. For our base case we will examine the case where $f(x_1, x_2, \ldots, x_n) = x_i$. Then $r_1 \leftarrow r_1 + r_2 \cdot f(x_1, x_2 \ldots x_n)$ implies that $r_1 = x_i$ which is our desired result. For our inductive step, we examine two cases, that case that we com-

bine two sub-formulas $g(x_1, x_2, \ldots x_n)$ and $h(x_1, x_2, \ldots x_n)$ under addition and under multiplication. Case 1: $f(x_1, x_2, \ldots, x_n) = g(x_1, x_2, \ldots, x_n) + h(x_1, x_2, \ldots, x_n)$

Let us preform the following operations:

$r_1 \leftarrow r_1 + r_2 \cdot g(x_1, x_2, \ldots, x_n)$
$r_1 \leftarrow r_1 + r_2 \cdot h(x_1, x_2, \ldots, x_n)$

With initial values of $r_1 = 0$, $r_2 = 1$, the first line will place the value of g into r1, and the second line will add the value of h into r_1, which equals g. Thus by the end of the assignment, r_1 holds the value of g + h, which is $f(x_1, x_2, \ldots, x_n)$, which was our desired outcome.

Case 2: $f(x_1, x_2, \ldots, x_n) = g(x_1, x_2, \ldots, x_n) \cdot h(x_1, x_2, \ldots, x_n)$

We start be preforming the following operations:

$r_3 \leftarrow r_3 + r_2 * g(x_1, x_2, \ldots, x_n)$
$r_1 \leftarrow r_1 + r_3 * h(x_1, x_2, \ldots, x_n)$

Note that at this point, r_1 holds the value we it to hold, namely $f(x_1, x_2, \ldots, x_n)$, and thus the proof is complete. However we can take it a step further and show that by preforming

$r_3 \leftarrow r_3 - r_2 \cdot g(x_1, x_2, \ldots, x_n)$
$r_1 \leftarrow r_1 - r_3 \cdot h(x_1, x_2, \ldots, x_n)$

you can restore the previous value of r_1 and r_3.

Recall that in that the original problem statement one of our conditions were that our Hard Drive was full, and if we were to use it, we could not compromise any data on it. More over, the length of the program will be $\leq 4^d$ where d is the depth of the tree. We can guarantee the restoration due to the fact that we require our input form a ring structure.

We can use the above to solve our original problem by looking at the Ring of nxn matrices over GF_2, and building a binary tree of matrices in

a similar manner to the previous problem. In this case we would initialize r_1, *and* r_3 to be the 0 matrix, and r_2 to be the appropriately sized identity matrix, I_n. The overall running time for the approach is approximately $O(n^4)$. In total we used O(log(n))-bits of RAM, and $O(n^2)$ bits of Hard Drive.

Planar Graph Drawing (lecture by Partice Ossona de Mendez)

Dylan Quintana

Suppose we have some planar graph G represented in computer memory. We would like to have an algorithm to draw G in its planar form; that is, with no crossing edges. As it turns out, there is an algorithm that can do this in linear time that relies on a depth-fist search. However, it is much too complicated to explain here, so instead we will focus on interesting ways of representing special cases of planar graphs.

1 2-Connected Graphs

Definition 1.1. A *2-connected graph* is a connected graph that requires the removal of at least two vertices to become disconnected.

Consider a 2-connected planar graph G. It is possible to direct the edges of G to form a graph G' that has a unique source node s with indegree 0 and sink node t with outdegree 0; this is known as a *bipolar orientation* of G. As a consequence of this, every vertex of G' must lie on at least one path from s to t. This places some constraints on the structure of the graph. For instance, every cycle in G must have one group of consecutive edges directed in one orientation and the rest of the edges directed in the other orientation. Otherwise, we would violate either planarity or uniqueness of the source and sink. Similarly, if we consider the orientations of the edges incident with a particular vertex, it must be the case that all of the outward edges are consecutive and all of the inward edges are consecutive (going around the vertex) Figure 1.

We can define two partial orders on the graph G'. If e and f are two edges on a common path from the source to the sink, we say that $e < f$ if e comes before f on the path. If e and f are two edges that do not share a path from the source to the sink, we say that $e <^* f$ if e is to the left of f on a planar representation of G' (there is a precise mathematical definition of what "to the left" means in this context). Based on these two partial orders, we can represent G' in a nice way by having every vertex be a horizontally enlongated rectangle, with incoming edges entering the bottom and outgoing edges exiting the top Figure 2.

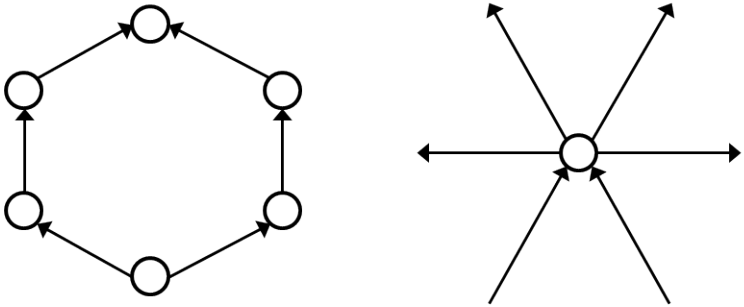


Figure 1: Allowed orientations of edges of G' in cycles and around single vertices

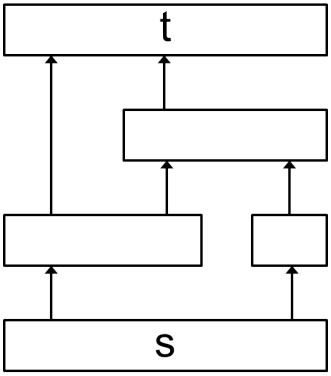


Figure 2: A planar graph with vertices drawn as boxes.

Given this planar representation of the graph G', we can create a dual graph G^* as follows: every face of G' will become a vertex of G^*, and two vertices of G^* will have an edge between them if their faces in G' are adjacent. The dual graph will have a new source node s' to the left of G' and sink node t' to the right of G'. The edges of G^* will be directed from the source to the sink, from left to right. Such a construction is shown in Figures 3 and 4.

If G is two-colorable, we can create a representation of G in which the vertices are represented as horizontal or vertical line segments based on their

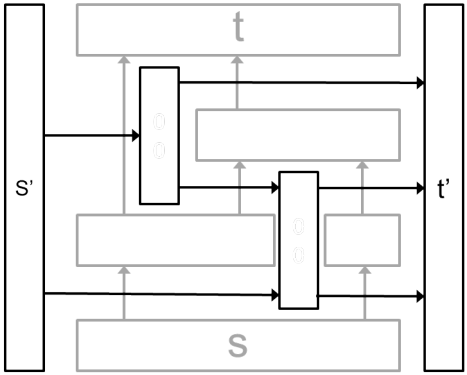


Figure 3: The construction of the dual graph. The gray vertices and edges belong to the orignal graph G', the black to the dual graph G^*.

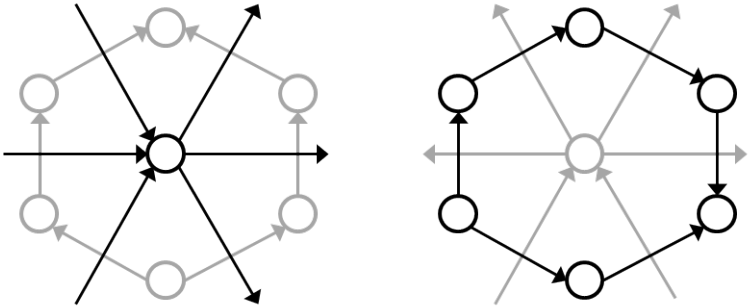


Figure 4: We find that G^* obeys the same structural constraints as G', because the cycle and single vertex orientation restrictions are in fact duals of each other (original graph in gray, dual in black).

color, and edges are formed between line segments that meet at a corner Figure 5.

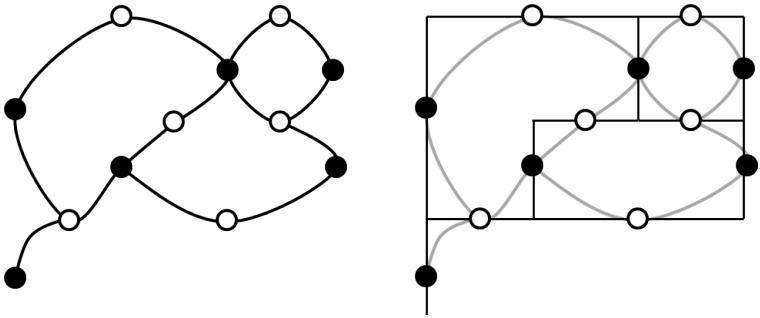


Figure 5: A two-colorable planar graph (left) and its representation as a set of perpendicular line segments (right).

2 Packing

Definition 2.1. A *bounded indegree orientation* is a way of directing the edges in a graph such that no vertex has an indegree larger than some specified constant.

By Euler's formula, a planar graph satisfies $e \leq 3v - 6$, which suggests it might be possible to orient the edges of any planar graph in such a way that the indegree of each vertex is bounded by 3. This is indeed possible, and we can find such an orientation through a process called *packing*. We begin by adding edges to the graph until it becomes maximal, containing only triangular faces. We then draw the graph in a planar form with a set of three adjacent vertices forming a large upward-pointing triangle with a horizontal base, and the remainder of the vertices lying within this triangle. We number the two outer vertices at the base of the triangle 1 and 2. An example of the result is shown in Figure 6.

The next step is performing the following process until all vertices are numbered:

1. Choose a vertex v that is adjacent to at least two numbered vertices such that the face F formed by v and the all numbered vertices it is adjacent to has no other vertices in its interior.

2. Orient toward v the two edges connecting v to the two vertices it is adjacent to in the cycle that forms the perimeter of F.

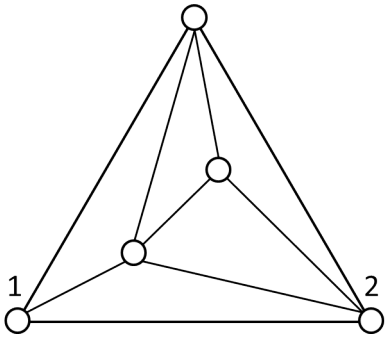


Figure 6: A maximal planar graph in a triangular representation at the start of the packing algorithm.

3. Orient outward from v all other edges that connect v to a numbered vertex.

4. Number v with the next unused natural number.

This algorithm ensures that all vertices will have an indegree of 3 or lower: two edges are oriented toward each vertex v when it is numbered, and a maximum of one edge can be oriented toward v when a later vertex is numbered, because once one such edge is added, v will be in the interior of the subgraph induced by all numbered vertices Figure 7. As a final step, we remove the edges we added at the beginning to make the graph maximal, which can of course only decrease the indegree of each vertex.

When the packing process is complete, the graph will have a nice orientation where the indegree of each vertex is bounded by 3, which can be used to represent the graph in different ways. For instance, numbering the vertices and labeling the edges in this manner is the first step in an algorithm for representing the graph with vertices in a $2n \times 2n$ grid of equally spaced points and straight edges. The vertex numbers also allow us to represent the vertices of the graph as a set of triangles where two triangles touch if the vertices are adjacent in the graph Figure 8.

As a final interesting note, it can be shown that the leftward, rightward, and downward-oriented edges in the maximal graph each form a tree that spans all vertices except two of the three outer corners.

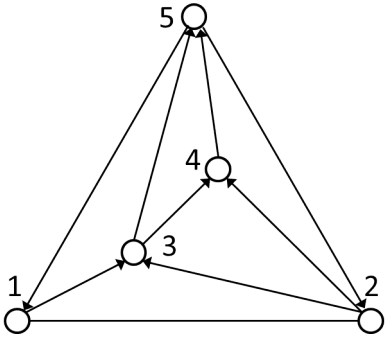


Figure 7: The planar graph from Figure 6 with its edges directed in an orientation with a bounded indegree of 3, with vertices numbered using the packing algorithm. Either orientation can be chosen for the bottom edge.

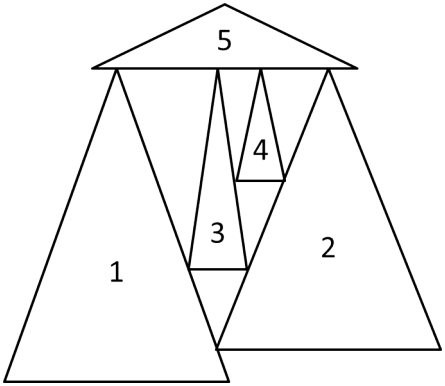


Figure 8: A represenation of the planar graph from Figure 7 where the vertices are isosceles triangles.

Hamiltonian Paths on Interval Graphs (lecture by Jiří Fiala)

Zach J. Wheeler

1 Overview

Professor Fiala first noted that the first few days of morning lecture were intended to give us a little additional background in discrete math, with a sampling from different topics. Professor Fiala himself presented a few solved problems, and also suggested a few open problems, all focused on the subject of Hamiltonian paths on interval graphs. The style of the presentation was very interactive.

2 Interval Graphs

2.1 Hamiltonian Paths

Given a collection of intervals in $\mathbb{R}$, can the intervals be arranged into a sequence such that consecutive intervals intersect? If we define an interval graph to be a vertex set of intervals in $\mathbb{R}$ with edges between intervals that intersect, the problem becomes finding a Hamiltonian path on an interval graph.

Note that if some intervals are included in others, it is possible for the graph to have a claw (i.e. a $K_{1,3}$ subgraph) that precludes a Hamiltonian path.

A known necessary condition for the existence of a Hamiltonian path in any graph is the following: for any $S \subseteq V$ we must have $\#comp(G$ $S) \leq |S|+1$, where $\#comp(G)$ denotes the number of connected components of G. This can be seen by taking an arbitary $S \subseteq V$ and considering it in the context of a Hamiltonian path $P = (p_0, p_1, ..., p_n)$ on the same graph: for any $p_i, p_j \in S$, with $i < j$, the subgraph of G induced by $p_{i+1}, p_{i+2}, ..., p_{j-1}$ must be connected, else P is not a path. This condition is not sufficient in general. However, is it sufficient for interval graphs?

2.2 Monotone Hamiltonian Paths

An additional restriction we could apply to Hamiltonian paths may be observed by considering each interval to represent the start and end time of some event. Suppose we wished to spend some time visiting each event exactly once; in this case, we cannot return to an earlier time, we are forced to proceed only forward. We call such a path monotone. If an interval graph G has a Hamiltonian path P, is it possible to rearrange P so it becomes monotone?

At this point we were left to tackle these two questions, which at first seemed unrelated, but as we discovered later, the second is an intermediate step toward solving the first. Two hints were given for this second problem: one was that we should look at the "times" that the path transfers between intervals and hold those times fixed, and the other was that we should endeavor to simplify the problem before attempting a proof.

2.3 Simplification Structure

We made little progress in the time we were left, but we did succeed in clarifying the second problem to our satisfaction, establishing a more rigorous definition. The only way is to consider the transfer times between intervals, and require these to be non-decreasing. We decided that unlike the general Hamiltonian path problem on interval graphs, it is probably not helpful to think of the monotone Hamiltonian path in terms of a purely graph abstraction.

Professor Fiala returned and guided us toward finding a way we could simplify an interval graph while still retaining exactly the same monotone paths. In essence, we project the interval endpoints to integers, and contract the lengths as much as possible without creating or destroying intersections between any pair of intervals. If we stack the intervals as shown below and draw vertical lines where interval endpoints appear, we can constrain the transfer times to places where vertical lines are drawn, which also correspond to maximal cliques in the interval graph.

If an algorithm can be found to compute a monotone Hamiltonian path, then the Hamiltonian path problem is solved on interval graphs if every

graph that has a Hamiltonian path also has a monotone Hamiltonian path. We were beginning to see the connection.

2.4 Constructing a Monotone Hamiltonian Path

We were left to work by ourselves again, and again we made little progress. We tried several approaches to both problems, but none quite worked, so it was necessary for Professor Fiala to finally show us. As it turned out, we were nearly there. The solution to constructing a monotone Hamiltonian path is given below.

Note that any Hamiltonian path will be piecewise monotone. So, first consider the case where we want to merge two monotone paths P and Q that both begin at the same interval u which is leftmost in both P and Q. This is equivalent to merging one path that changes direction from left to right at u. (Note the argument is symetric if we reverse left and right.) Next sort the intervals in the order of the time the path transfers to another interval, and call this sorted list $L = (p_1, p_2, ..., p_n)$, denoting the transfer times $T = (t_1, t_2, ..., t_n)$. There are two special cases; for u, the path leaves twice, so we ignore the earlier time; for the last interval in each monotone path, the path never leaves, so we consider the end of the interval to be the time the path leaves. Now consider any pair of intervals in L, p_i and p_{i+1}, where one is in P and the other is in Q. WLOG $p_i \in P$ and $p_{i+1} \in Q$, and $t_i \leq t_{i+1}$. Suppose p_i transfers to x at time t_i. Since P and Q both start at u, and t_i and t_{i+1} are adjacent in T, it must be possible to transfer from p_i to p_{i+1} instead of x at t_i. A simple induction argument shows that applying this alteration to every such pair will create a monotone Hamiltonian path R through all the intervals in P and Q.

To merge three monotone paths P, Q, and S, which WLOG are monotone in the directions left, right, and left respectively, we first construct R from P and Q as before. This gives us two monotone Hamiltonian paths which do not begin at the same interval. However, we can simply choose some interval u in R which overlaps the endpoint of S that joined S to Q, cut R at u into R' on the left and R'' on the right, so that S and R'' both begin with leftmost u, merge the two paths as before, and then simply concatenate R' with the resulting path. Actually it is not quite that simple, but the precise argument is very complex, and the idea is the same, so Professor Fiala skipped the details.

The two methods given above allow us to construct a monotone Hamiltonian path from any Hamiltonian path inductively, so it follows that every

interval graph that has a Hamiltonian path also has a monotone Hamiltonian path.

2.5 Algorithm to Find Monotone Hamiltonian paths

A proof was not given – the day was drawing to an end, and Professor Fiala could see we were feeling the effects of jet lag – but essentially, once the set of intervals has been simplified as specified before in section 2.3, it is enough to sweep from left to right, choosing as the next interval whichever is shortest among those that can be reached. This together with the above proof solves the Hamiltonian path problem for interval graphs with a polynomial time algorithm.

Additionally, it was mentioned that the inequality from section 2.1 is indeed sufficient for interval graphs, but this again was not proved.

3 Open Problems

Along the way, three open problems were mentioned. They are given below:

1. A game can be defined on an interval graph where players alternate turns choosing from a collection of intervals to extend a path until it is not possible to extend it further. What are the winning strategies? A few variants of the game are possible. We thought briefly about this problem, and concluded that it falls under the Sprague-Grundy theorem for impartial games.

2. In the case of a Hamiltonian cycle, is it possible to choose a "leftmost" and "rightmost" interval such that the two paths between them can be made monotone? How can these intervals be chosen?

3. What can we say about Hamiltonian paths in interval graphs where one or both of the endpoints are fixed? More is known about this question than the others.

3-Colorings and 4-Critical Graphs (lecture by Zdeněk Dvořák)

Elizabeth Yang

1 Introduction

Consider the 3-coloring of a graph G. Determining whether or not a graph is 3-colorable is interesting from an algorithmic point of view, however it has been shown that this problem is NP-hard. Instead, we discuss 4-critical graphs, the theorem of Kostochka and Yancey, its proof, and its applications.

1.1 Definition

A graph G is **4-critical** if G is not 3-colorable, but any proper subgraph of G is.

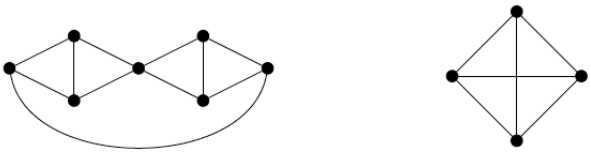


Figure 1: These are two examples of 4-critical graphs.

1.2 Preliminary Results

Proposition 1.1. *If a graph G is 4-critical, the minimum degree of G is at least 3.*

Proof. Assume for contradiction that there exists a vertex v of degree at most 2. Since G is 4-critical, $G \setminus v$ is 3-colorable. v has at most 2 neighbors, so we do not need a fourth color for v. G is then 3-colorable as well, a contradiction. □

Corollary 1.2. *Given a 4-critical graph G, $|E(G)| \geq \frac{3|V(G)|}{2}$.*

Proof. We know that $2 \cdot |E(G)| = \sum_{v \in V(G)} \deg(v)$. Since $\deg(v) \geq 3$ for all $v \in V(G)$, $|E(G)| \geq \frac{3|V(G)|}{2}$ as desired. □

Note that the bound from Corollary 1.2 is tight; equality is achieved by K_4. K_4 is the only 4-critical graph with this property, to be proven below.

Theorem 1.3. (Brooks) *Let Δ be the maximum degree of connected graph G. If $G \neq K_{\Delta+1}$ and G is not an odd cycle, then G can be colored with Δ colors.*

Corollary 1.4. *If graph G is 4-critical and $\Delta = 3$, G must be K_4.*

Proof. Since G cannot be colored with Δ colors, G must either be K_4 or an odd cycle. $\Delta = 3$ implies G must be K_4. □

2 Statement of Theorem

Kostochka and Yancey were able to improve the bounds from Corollary 1.2.

Theorem 2.1. (Kostochka, Yancey) *If graph G is 4-critical,*

$$|E(G)| \geq \frac{5|V(G)| - 2}{3}$$

.

We will next outline the proof.

3 Proof of Theorem

3.1 The Potential Function

Given graph G, define its potential function:

$$P(G) = 5|V(G)| - 3|E(G)|$$

Proving the bound from the theorem is equivalent to showing that $P(G) \leq 2$.

3.2 Proof Lemmas

For contradiction, assume there exists some 4-critical graph G with $P(G) \geq 3$. Choose G with the minimum $|V(G)| + |E(G)|$.

Lemma 3.1. *Let H be a subgraph of G. Then:*

(i) $P(H) \geq 3$ *if* $H = G$.

(ii) $P(H) = 5$ *if H is a single vertex.*

(iii) $P(H) \geq 6$ *otherwise.*

Proof. The first two statements are easily verified. Choose subgraph H such that $H \neq G$ and H is not a single vertex. If H is the subgraph of a triangle, we can easily show that $P(H) \geq 5$.

We may assume H is not the subgraph of a triangle. Since H is a proper subgraph of G, it has a 3-coloring. Condense all vertices in $V(H)$ of the same color to a single vertex. We have condensed H into triangle T.

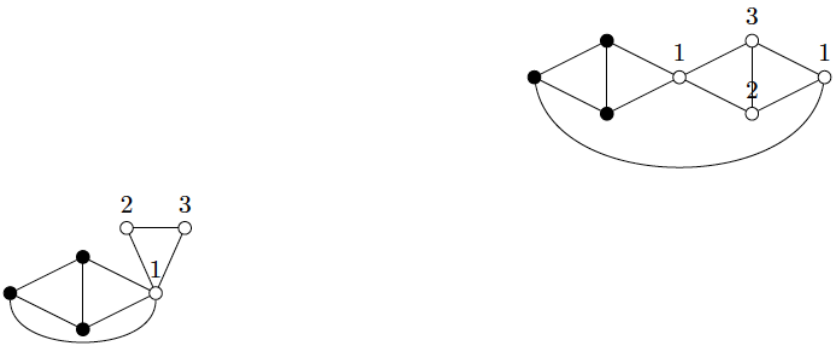


Figure 2: The left shows G, with H highlighted. The right shows G_1.

Let G_1 be the graph obtained after condensing H into T in G. G_1 cannot be 3-colorable; otherwise, we can expand T back into H and obtain a 3-coloring for G. Thus, G_1 contains a 4-critical subgraph G_2. Replace $G_2 \cap T$ with H to obtain G_3.

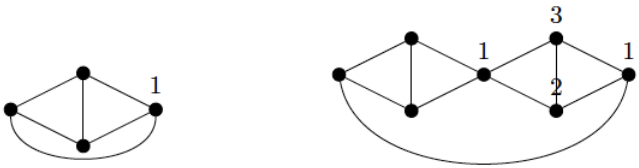


Figure 3: G_2 is on the left, G_3 on the right. In this particular case, $G_3 = G$.

We now obtain the expression $P(G_3) = P(G_2) - P(G_2 \cap T) + P(H)$. By minimality of G, $P(G_2) \leq 2$. Since $G_2 \cap T$ is the subgraph of a triangle, $P(G_2 \cap T) \geq 5$. Therefore, $P(G_3) \leq P(H) - 3$.

Iterate this process, treating G_3 as the new H to get G_3'. (G_3 is indeed a subgraph of G.) Stop once the entirety of G is restored. The iteration gives us the following chain of inequalities:

$$\begin{aligned} P(G_3) &\leq P(H) - 3 \\ P(G_3') &\leq P(G_3) - 3 \\ P(G_3'') &\leq P(G_3') - 3 \\ &\vdots \\ P(G) &\leq P(G_3^{(k)}) - 3 \end{aligned}$$

Here, k represents the number of iterations to get to G. From the system, we have $P(G) \leq P(H) - 3k$. Since $k \geq 1$ and $P(G) \geq 3$ by assumption, $P(H) \geq 6$. □

The following lemma is a corollary of Lemma 3.1.

Lemma 3.2. *K_4 without an edge ($K_4 \setminus e$) is not a subgraph of G.*

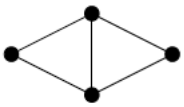


Figure 4: $K_4 \setminus e$, the forbidden subgraph.

Proof. $P(K_4 \setminus e) = 5$, but $K_4 \setminus e$ is not a single vertex. □

Lemma 3.3. *Let H be a proper subgraph of G with $H \neq G \setminus e$. Consider vertices $u, v \in H$. The graph $H + uv$ is 3-colorable.*

Proof. Suppose for contradiction that $H + uv$ is not 3-colorable. Then, $H + uv$ contains a 4-critical subgraph, H'. Since H' is smaller than G, $P(H') \leq 2$. Since H is a subgraph of G, $P(H) \geq 6$. However, $P(H') = P(H + uv) = P(H) - 3$, so $P(H') \geq 3$, a contradiction. □

Lemmas 3.2 and 3.3 help us make an improvement on Lemma 3.1.

Lemma 3.4. *Let H be a subgraph of G. Then:*

(i) $P(H) \geq 3$ *if* $H = G$.

(ii) $P(H) = 5$ *if H is a single vertex.*

(iii) $P(H) = 6$ *if H is a triangle.*

(iv) $P(H) \geq 6$ *if* $H = G \setminus e$.

(v) $P(H) = 7$ *otherwise.*

Proof. The first four statements can be easily verified. The proof of the last statement follows in a similar way from the proof of Lemma 3.1.

Choose H a proper subgraph of G with $H \neq G \setminus e$. We may also assume H is not a subgraph of a triangle. Take 2 vertices in H, x and y, which have neighbors in $V(G \setminus H$. Call the edges from x and y (out of H) e_1 and e_2, respectively.

Consider $H + xy$. By Lemma 3.3, $H + xy$ is 3-colorable, with x and y guaranteed different colors. We proceed with the proof of Lemma 3.1, first contracting H into triangle T. After obtaining graph G_3, we have the inequality $P(H) \geq P(G_3) + 3$. We now have two cases:

- If $G_3 \neq G$, $P(G_3) \geq 6$ because G_3 is a proper subgraph of G, so $P(H) \geq 8$.
- If $G_3 = G$, edges e_1 and e_2 belong to $E(G_3)$. These edges also belong to $E(G_2)$. G_2 also contains at least 2 vertices of T because e_1 and e_2 have two different ends in T, so $P(T \cap G_2) \geq 6$. Therefore, $P(H) \geq P(G_3) + 4$ and $P(H) \geq 7$.

□

Lemma 3.4 helps us discover some more forbiddens subgraphs.

Lemma 3.5. *G does not contain 2 vertices of degree 3 in the same triangle.*

Proof. Assume for contradiction that G contains 2 vertices of degree 3 in the same triangle. Let these vertices be u and v. u and v share a neighbor w, but cannot share any other neighbor by Lemma 3.2. Let graph G_1 be obtained from G by deleting u and v and adding an edge e between two neighbors of u and v that are not equal to w.

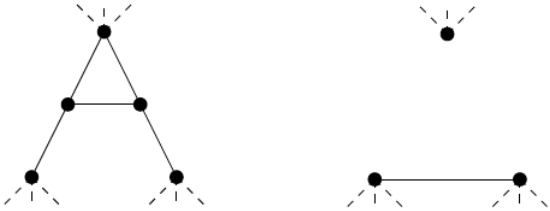


Figure 5: The forbidden subgraph in G and the same configuration in G_1.

Any 3-coloring of G_1 would extend to a 3-coloring of G, so G_1 cannot have a 3-coloring. G_1 therefore has a 4-critical subgraph, G_2. G_2 contains edge e; otherwise, G_2 is a subgraph of G. Next, obtain graph G_3 by restoring vertices u and v, but not joining them to w.

Figure 6: The forbidden subgraph in G_2 and the same configuration in G_3.

This tells us that $P(G_3) = P(G_2) + (5 \cdot 2 - 3 \cdot 2) = P(G_2) + 4$. Since G_2 is smaller than G, $P(G_2) \leq 2$, so $P(G_3) \leq 6$, contradicting Lemma 3.4. □

Lemma 3.6. *If u and v are adjacent vertices of degree 3, then both u and v are contained in some triangles.*

Proof. Without loss of generality, let u not be part of any triangle, so its neighbors do not have an edge between them. Perform the following reduction to graph G_1. Remove u and v and condense u's neighbors into a single vertex.

Figure 7: The forbidden subgraph in G and the same configuration in G_1.

Any 3-coloring of G_1 would extend to a 3-coloring of G, so G_1 cannot be 3-colorable. G_1 must have a 4-critical subgraph G_2. G_2 must contain the condensed vertex, otherwise, G_2 would be a subgraph of G. Obtain the graph G_3 by restoring u but not v. G_3 is a subgraph of G.

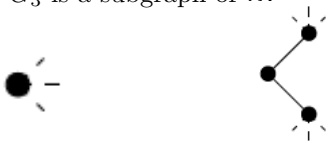
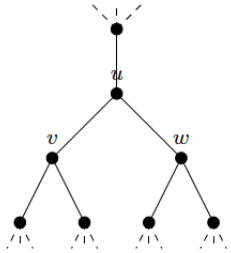


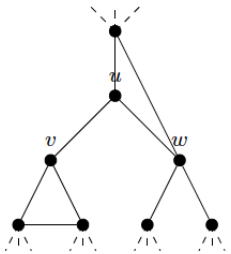
Figure 8: The forbidden subgraph in G_2 and the same configuration in G_3.

This tells us $P(G_3) = P(G_2) + (5 \cdot 2 - 3 \cdot 2) \leq 6$, a contradiction to Lemma 3.4. □

Lemma 3.7. *Every vertex of degree 3 has at most one neighbor of degree 3.*

Proof. Assume for contradiction there exist some vertex of degree 3, u, with two neighbors of degree 3, v and w. Lemma 3.6 says u and v must belong in triangles, so there exists an edge between the neighbors of u and the neighbors of v. The triangle containing u now also contains w. However, u and w now contradict Lemma 3.5. □





We can now proceed to the actual proof. The proof uses a double-counting method.

3.3 Proof

Assign initial charges to the vertices according to the following charge function: $\mathrm{ch}_0(v) = 10 - 3\deg(v)$. Then, we compute the total charge:

$$\begin{aligned}\sum_{v\in V(G)} \mathrm{ch}_0(v) &= \sum_{v\in V(G)} [10 - \deg(v)] \\ &= 10|V(G)| - 3\sum_{v\in V(G)} \deg(v) \\ &= 10|V(G)| - 6|E(G)| \\ &= 2P(G) \geq 6\end{aligned}$$

Next, we discharge the graph by sending a charge of $\frac{1}{2}$ from each vertex of degree 3 to each neighbor of degree at least 4. Let $\mathrm{ch}(v)$ represent charge after discharging.

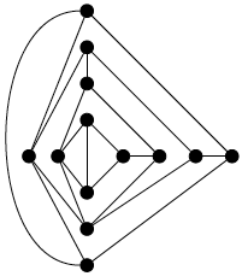
For vertices of degree 3, $\mathrm{ch}(v) \leq 0$ by Lemma 3.7. For vertices of degree at least 4, we can perform the following computation, where we assume the worst case scenario (all neighbors degree 3):

$$\begin{aligned}\mathrm{ch}(v) &\leq \mathrm{ch}_0(v) + \frac{\deg(v)}{2} \\ &\leq 10 - 3\deg(v) + \frac{\deg(v)}{2} \\ &=\leq 10 - \frac{5\deg(v)}{2} \leq 10 - 10 = 0\end{aligned}$$

This computation also confirms that $\text{ch}(v) \leq 0$ for all other vertices. It follows that $\sum_{v \in V(G)} \text{ch}(v) \leq 0$, a contradiction to the initial charge computation.

3.4 Tightness of Bound

$P(G) \leq 2$ is a tight bound for $|V(G)|$ arbitrarily large. Below is an example of one graph in a family of graphs that satisfy $P(G) = 2$.



4 Applications of Theorem

We can apply the bound to prove the following theorem:

Theorem 4.1. (Grotzsch) *Any planar, triangle-free graph is* 3*-colorable.*

Proof. Assume for contradiction that there exist planar, triangle-free graphs that are not 3-colorable. Choose G such that $|V(G)| + E(G)|$ is the minimum. Observe that G must be 4-critical. We now have two cases:

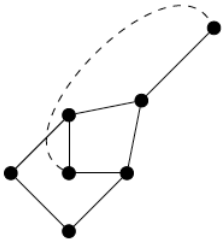
- G has no 4-faces. In this case, we apply Euler's formula and that $|F(G| \leq \frac{2|E|}{5}$:

$$\begin{aligned} |E(G)| + 2 &= |V(G)| + |F(G)| \\ &\leq |V(G)| + \frac{2|E(G)|}{5} \\ \frac{3|E(G)|}{5} &\leq |V| - 2 \\ |E(G)| &\leq \frac{5|V(G)| - 10}{3} \end{aligned}$$

This contradicts our theorem and G cannot be 4-critical.

- G has a 4-face. The idea is to identify a pair of opposite vertices of the 4-face and consolidate them, forming graph G'. G' cannot have a 3-coloring, otherwise it extends to a 3-coloring of G.

  The only time we may run into an issue is when for both pairs of opposite vertices in the 4-face, a triangle forms when consolidated. However, this may not happen; we obtain a contradiction to the planarity of G. See the figure below.



  We are then reduced to the first case and we are done.

□

4.1 Extensions and Generalizations

We can establish similar bounds for 5-critical graphs. We can attempt to generalize our proof for 4-critical graphs using the potential function $9|V(G)| - 4|E(G)| \leq 5$.

These bounds have already been generalized for k-critical graphs.

Hardness of the L-bounded cut

Pavel Dvořák and Dušan Knop

In this part we prove MLBC parametrized by path-width is W[1]-hard by FPT-reduction from k-Multicolor Clique.

PROBLEM:	k-Multicolor Clique
Instance:	k-partite graph $G = (V_1 \dot\cup V_2 \dot\cup \ldots \dot\cup V_k, E)$, where V_i is independent set for every i and they are pairwise disjoint
Parameter:	k
Goal:	find a clique of size k

Denoting Sets $V_1, \ldots, V_k$ are always the color class of G. We denote edges between V_i and V_j by E_{ij}. The problem is W[1]-hard even if every independent set V_i has same size and the number of edges between every V_i and V_j is same. In whole part we denote the size of arbitrary V_i by N and size of arbitrary E_{ij} by M. For FPT-reduction from k-Multicolor Clique to MLBC we need:

1. Create a MLBC instance $G' = (V', E'), s, t, L$ from k-Multicolor Clique instance $G = (V_1 \dot\cup V_2 \dot\cup \ldots \dot\cup V_k, E)$ where size of G' is polynomial from the size of G.
2. Prove that G contains k-clique if and only if G' contains L-bounded cut of size $f(k, N, M)$ where f is polynomial.
3. Prove that path-width of H is smaller than $g(k)$ where g is computable function.

Our ideas were inspired by work of Michael Dom et al. [1]. They proved W[1] hardness of Capacitated Vertex Cover and Capacitated Dominating Set parametrized tree width of input graph. We remarked that their reduction also proves W[1] hardness of these problems parametrized by path-width.

1 Basic gadget

In k-Multicolor Clique problem we need select exactly one vertex from each independent set V_i and exactly one edge from each E_{ij}. And we have

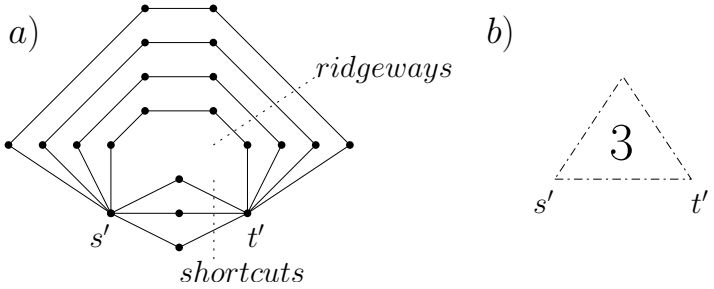


Figure 1: a) Example of butte for $h = 3$ and $Q = 4$. b) Simply diagram for butte of height 3.

to make certain that if $e \in E_{ij}$ is the selected edge and $u \in V_i, v \in V_j$ are the selected vertices then $e = \{u, v\}$. The idea of reduction is to have a basic gadgets for every vertex and edge. Let g_v be the gadget for v. We connect gadgets g_v for every v in V_i into a path P_i. The path P_i is cut in gadget $g(v)$ if and only if the vertex $v \in V_i$ is selected into clique. The same idea will be used for selecting the edges.

Definition 1.1. Let $h, Q \in \mathbb{N}$. Butte $B(s', t', h, Q)$ is graph which contains h paths of length 2 and Q paths of length $h+2$ between s' and t'. The short paths (of length 2) are called shortcuts, the long paths are called ridgeways and the parameter h is called height.

Butte for $h = 3, Q = 4$ is shown in Figure 1 part a. In our reduction all buttes will have the same parameter Q (it will be computed later). For simplicity we depicts butte as a dash dotted line triangle with its height h inside (see Figure 1 part b), or only as a triangle without the height if it is not important. Let $B(s', t', h, Q)$ be a butte. We denote by $s(B), t(B), h(B), Q(B)$ the parameters of butte B s', t', h and Q, respectively.

Observation 1.2. *Path-width of arbitrary butte B is at most 3.*

Proof. If we remove vertices $s(B)$ and $t(B)$ from B we get $Q(B)$ paths from ridgeways and $h(B)$ isolated vertices from shortcuts. This graph has certainly path-width 1. If we add $s(B)$ and $t(B)$ to every node of the path decomposition we get proper path decomposition of B with width 3. □

Let $B(s', t', h, Q)$ be a butte. Let P_{uv} be the shortest path between u

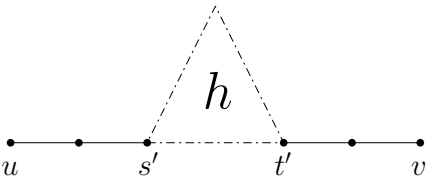


Figure 2: The example of path going through a butte.

and v, which enters into B in s' and leaves it in t' (see Figure 2). The important properties of the butte B are:

1. By removing one edge from all h shortcuts of butte B, we extend path P_{uv} by h. If we cut all shortcuts of butte B we say the butte B is ridged.

2. If the size of the cut is bounded by $K \in \mathbb{N}$ we can increase $Q > K$. Therefore P_{uv} cannot be cut by removing edges from B (only extended by ridging the butte B).

3. Butte B has constant path-width independent on h and Q.

2 Butte path

In this section we define how we connect buttes into a path, which we call highland. The main idea is to have highland for every $i \neq j$. In highland for $i \neq j$, there are buttes for every vertex $v \in V_i$ and every edge $e \in E_{i,j}$. First we connect vertex buttes into the path and after them we connect edge buttes. Then we set the butte heights and limit the size of the cut in such way that exactly one vertex butte and exactly one edge butte have to be ridged. And if buttes for vertex v is ridged, then only buttes for edges incident with v can be ridged. Thus, vertex $v \in V_i$ will be chosen to clique if and only if butte for v will be ridged. Formal description of highland is in the following definition.

Definition 2.1. Highland $H(X, Y, s, t)$ is a graph containing 2 vertices s and t and $Z = X + Y$ buttes $B_1, \ldots, B_Z$ where:

1. $s = s(B_1), t = t(B_Z)$ and $t(B_i) = s(B_{i+1})$ for every $1 \leq i < Z$.

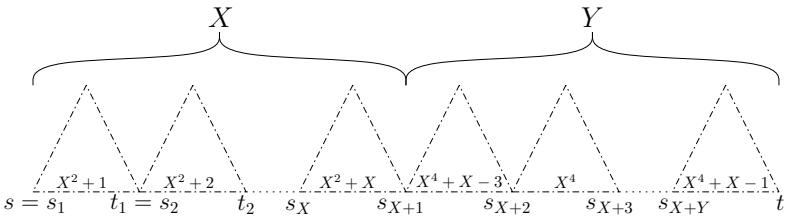


Figure 3: The example of highland $H(X, Y, s, t)$.

2. $h(B_i) = X^2 + i$ for $1 \leq i \leq X$.
3. $h(B_i) \in \{X^4, \ldots, X^4 + X - 1\}$ for $X + 1 \leq i \leq Z$.
4. $Q(B_i) = X^4 + X^2$ for every i.

Let $H(X, Y, s, t)$ be highland. We call buttes $B_1, \ldots, B_X$ from H low and buttes $B_{X+1}, \ldots, B_{X+Y}$ high (low buttes will be used for vertices and high buttes for edges). The vertex $t(B_X) = s(B_{X+1})$, where low and high buttes meet, is called the center of highland H. Note that there can be more buttes with the same height among high buttes and they are not ordered by height as the low buttes. The example of highland is shown in Figure 3.

Proposition 2.2. *Let $H(X, Y, s, t)$ be a highland. Let $L = 2(X + Y) + X^4 + X^2 + X - 1$. Let C be the L-cut of size $X^4 + X^2 + X$, which cut all path of length L and shorter between s and t then:*

1. *The cut C contains only edges obtained by ridging the exactly two buttes B_i, B_j, such that B_i is low and B_j is high.*
2. *Let B_i be the ridged low butte and B_j be the ridged high butte. Then, $h(B_j) = X^4 + X - i$.*

Proof. Every butte has at least $X + 1$ shortcuts and $X^4 + X^2$ ridgeways. Therefore, C can not cut all paths in H between s and t and it is useless to add edges from ridgeways to the cut C. Note that shortest st-path in H has length $2(X + Y)$.

1. If we ridge every low butte we extend the shortest st-path by $X^3 + \frac{X^2}{2} + \frac{X}{2}$. However, it is not enough and at least one high butte has to be ridged. Two high buttes from cannot be ridged otherwise the

cut would be bigger then the bound. No high butte can extend the shortest st-path enough, therefore at least one low butte has to be ridged. However, two low buttes and one high butte cannot be ridged because the cut C would be bigger then the bound.

2. Let F be the set of removed edges from ridged buttes B_i and B_j. Height of B_i is $X^2 + i$. Therefore the length of the shortest st-path after ridging B_i and B_j and the size of F is $2(X+Y)+X^2+i+h(B_j)$. If $h(B_j) < X^4 + X - i$ then shortest st-path is strictly shorter then $2(X+Y)+X^4+X^2+X$. Thus, F is not L-cut. If $h(B_j) > X^4+X-i$ then and $|F| > X^4 + X^2 + X$ thus F is bigger than C.

□

3 Reduction

In this section we present our reduction. Let $G = (V_1 \dot\cup V_2 \dot\cup \ldots \dot\cup V_k, E)$ be the input for k-Multicolor Clique. As we stated in the last section, the main idea is to have low butte B_v for every vertex v and high butte B_e for every edge e. Vertex v and edge e is selected into the k-clique if and only if butte B_v and butte B_e are ridged. From G we construct MLBC input G', s, t, L:

1. For every $1 \leq i, j \leq k, i \neq j$ we create highland $H^{i,j}(N, M, s, t)$ of buttes $B_1^{i,j}, \ldots, B_{N+M}^{i,j}$.

2. Let $V_i = \{v_1, \ldots, v_N\}$. Vertex $v_\ell \in V_i$ is represented by low butte $B_\ell^{i,j}$ of the highland $H^{i,j}$ for every $j \neq i$. Thus, we have $k-1$ copies of buttes (in different highlands) for every vertex. Hence, we need to be certain that only buttes representing the same vertex are ridged. Note that buttes representing the same vertex have the same height and the same distance from the vertex s.

3. Let $E_{ij} = \{e_1, \ldots, e_M\}, i < j$. Edge $e_\ell = \{u, v\} \in E_{ij} (u \in V_i, v \in V_j)$ is represented by high butte $B_{N+\ell}^{i,j}$ of the highland $H^{i,j}$ and by high butte $B_{N+\ell}^{j,i}$ of the highland $H^{j,i}$. Note that two buttes represented the same edge has same distance from the vertex s. Let h_i, h_j be the heights of buttes representing vertices u and v, respectively. We set the height of high buttes:

(a) $h(B^{i,j}_{N+\ell}) = N^4 + N - h_i$

(b) $h(B^{j,i}_{N+\ell}) = N^4 + N - h_j$

4. We add edge $\left\{t(B^{i,j}_\ell), t(B^{i,j+1}_\ell)\right\}$ for every $1 \le i \le k, 1 \le j < k, i \ne j$ and $1 \le \ell < N$.

5. We add paths of length $N-1$ connected $t(B^{i,j}_\ell)$ and $t(B^{j,i}_\ell)$ for every $1 \le i, j \le k, i \ne j$ and $1 \le \ell < M$.

6. $L = 2(N+M) + N^4 + N^2 + N - 1$

We call paths between highlands in Items 4 and 5 the valley paths.

Observation 3.1. *Graph G' has polynomial size from graph G.*

Theorem 3.2. *If graph G has a clique of size k then (G', s, t) has an L-cut of size $k(k-1)(N^4 + N^2 + N)$.*

Proof. Let G has a k-clique $\{v_1, \ldots, v_k\}$ where $v_i \in V_i$ for every i and $e_{ij} = \{v_i, v_j\} \in E_{ij}$. For every i we ridge all $k-1$ buttes representing the vertex v_i in G'. And for every $i < j$ we ridge both buttes representing the edge e_{ij}.

We claim that set of removed edges from ridged buttes forms the L-cut. Let $H^{i,j}$ be an arbitrary highland. There is no st-path shorter then L in $H^{i,j}$. Let $h(B_v) = N^2 + \ell$ where B_v is arbitrary butte representing the vertex v_i. By construction of G', the high butte representing the edge e_{ij} in $H^{i,j}$ has height $N^4 + N - \ell$. Thus, ridged buttes in $H^{i,j}$ extend the shortest st-path by $N^4 + N^2 + N$ and it has length $2(M+N) + N^4 + N^2 + N$. Buttes representing the vertex v_i have same height. Thus, path through the low buttes of highlands using some valley path is always longer then path going through low buttes of only one highland. Therefore, it is useless to use valley paths among low buttes for the shortest st-path.

Other situation is among high buttes because buttes representing the same edge have different heights. Butte B_v representing vertex v_i extend the shortest path at least by N^2+1. Butte B_e representing edge $e_{i,j}$ extend the shortest at least by N^4. However, if $h(B_v)+h(B_e) < N^4+N^2+N$ then B_v and B_e have to be in different highlands. Therefore, the st-path going through B_v and B_e has to use a valley path between high buttes, which has length $N-1$. And such st-path has length at least $2(N+M)+N^4+N^2+N$.

We remove N^4+N^2+N edges from each highland and there are $k(k-1)$ highlands in G'. Therefore, G' has L-cut of the size $k(k-1)(N^4 + N^2 + N)$. □

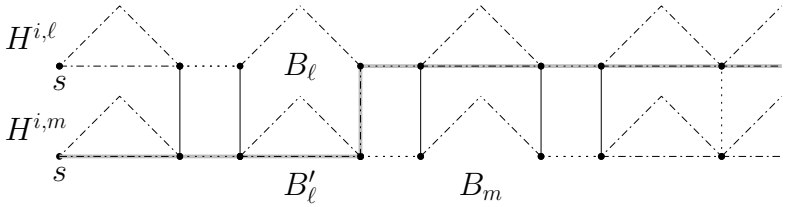


Figure 4: How to miss every ridged low butte if there are ridged two low buttes representing two different vertices from one color class. Ridged butte is depicted as triangle without hypotenuse.

Theorem 3.3. *If (G', s, t) has an L-cut of size $k(k-1)(N^4+N^2+N)$ then G has a clique of size k.*

Proof. Let C be an L-cut of G'. Every shortest st-path going through every highland has to be extended by N^4+N^2+N. By Proposition 2.2 (Item 1), exactly one low butte and exactly one high butte of each highland has to be ridged. We remove (N^4+N^2+N) from every highland in G'. Therefore, there can be only edges from ridged buttes in C.

For fixed i and $j \neq i$, highlands $H^{i,j}$ are the highlands which low buttes represent the vertex from V_i. We claim that ridged low buttes of $H^{i,1}, \ldots, H^{i,k}$ represent the same vertex. Suppose for contradiction, there exists two low ridged buttes B_ℓ of $H^{i,\ell}$ and B_m of $H^{i,m}$ which represent different vertex from V_i. Without loss of generality $H^{i,\ell}$ and $H^{i,m}$ are next to each other (i.e. $|\ell - m| = 1$) and distance from s to $s(B_\ell)$ is smaller than distance from s to $s(B_m)$. Let B'_ℓ be a butte of $H^{i,m}$ such that it has same distance from s as butte B_ℓ. The path s–$t(B'_\ell)$–$t(B_\ell)$–t (see Figure 4) does not go through any ridged low butte.

Therefore, this path is shorter than L, which is contradiction. We can use the same argument to show that there are not two high ridged buttes of highland $H^{i,j}$ and $H^{j,i}$ which represent different edges from E_{ij}.

We put into the k-clique $K \subset V(G)$ the vertex $v_i \in V_i$ if and only if arbitrary butte representing the vertex v_i is ridged. We proved in the previous paragraph that exactly one vertex from V_i can be put into the clique K. Let $e_{ij} \in E_{ij}$ is an edge represented by ridged buttes. We claim that $v_i \in e_{ij}$. Let $B \in H^{i,j}$ be a butte representing v_i with height $N^2+\ell$. Then by Proposition 2.2 (Item 2), butte $B' \in H^{i,j}$ of height $N^4+N-\ell$ has to be ridged. By construction of G', only buttes representing edges incident

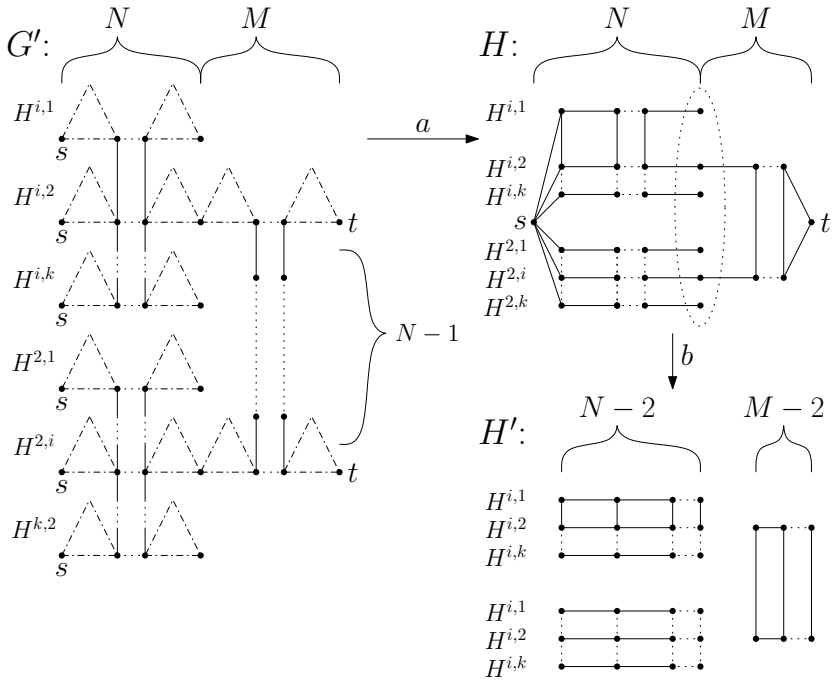


Figure 5: Transformation a replace all buttes in G' by single edges and contract long valley path into single edges. Transformation b removes vertices s and t and all highland centers (highlighted by dotted ellipsis) from H.

with v_i have such height. Therefore, chosen edges are incident with chosen vertices and they form the k-clique of the graph G. □

Observation 3.4. *Graph G' has path-width in $O(k^2)$.*

Proof. Let H be a graph created from G by replacing every butte by single edge and contract the valley paths between high buttes into single edges, see Figure 5 transformation a. Let U be a vertex set containing s, t and every highland center. Let H' be a graph created from H by removing all vertex from U, see Figure 5 transformation b.

Graph H' is unconnected and it contains k grids of size $(k-1)\times(N-2)$ and $\binom{k}{2}$ grids of size $2 \times (M-2)$. Path-width of $(k-1) \times (N-2)$ grids

is in $O(k)$, therefore $\mathrm{pw}(H') \in O(k)$. If we add set U to every node of a path decomposition of H' we get proper path decomposition of H. Since $|U| \in O(k^2)$, path-width of H is in $O(k^2)$. The edge subdivision does not increase path-width. Moreover, replacing edges by buttes does not increase it either (up to multiplication constant) because butte has constant path-width (Observation 1.2). Therefore, $\mathrm{pw}(G') = \mathrm{pw}(H) \in O(k^2)$.

□

Theorem 3.5. Minimal Length Bounded Cut *parametrized by path-width is* W[1]*-hard.*

Proof. The theorem is corollary of Observation 3.1 and 3.4 and Theorem 3.2 and 3.3. □

References

[1] Michael Dom, Daniel Lokshtanov, Saket Saurabh and Yngve Villanger *Capacitated Domination and Covering: A Parameterized Perspective*, Parameterized and Exact Computation, pages 78–90, 2008.

Hardness of $\mathcal{IV}$-matching

Lukáš Folwarczný, Dušan Knop[1]

Abstract

This text is just a preliminary version of our future article. $\mathcal{IV}$-matching is a generalization of perfect bipartite matching. The complexity of finding $\mathcal{IV}$-matching in a graph was posted as an open problem at ICALP 2014 conference. We give the proof that this problem is $\mathcal{NP}$-complete.

1 Introduction and Problem Definition

The generalization of perfect bipartite matching called $\mathcal{IV}$-matching is formulated in the full version of the article by Fiala, Klavík, Kratochvíl and Nedela [1].

Authors study algorithmic aspects of regular graph covers and one of their subproblems reduces to searching $\mathcal{IV}$-matching thus the authors ask the question whether there is an efficient algorithm solving this problem.

In this article we prove that the problem is $\mathcal{NP}$-complete and hence it is unlikely that there would an efficient algorithm.

Definition 1.1 ($\mathcal{IV}$-matching)**.** Let $G = (V, E)$ be a bipartite graph with following properties:

- Vertices are partitioned into sets $V_1, \ldots, V_\ell$ called *layers*. There are only edges between two consecutive layers V_k and V_{k+1} for $k = 1, \ldots, \ell - 1$.
- Each layer is further partitioned into *clusters*.
- Edges of G are described by edges on clusters; we call these edges *macroedges*. If there is a macroedge between two clusters, then vertices of these two clusters induce a complete bipartite graph. If there is no macroedge, then these vertices induce an edge-less graph.
- Macroedges between clusters of layers V_{2k} and V_{2k+1} form a matching (not necessarily a maximum matching).
- There are arbitrary macroedges between clusters of layers V_{2k-1} and V_{2k}.

[1]Author supported by the project SVV-2014-260103.

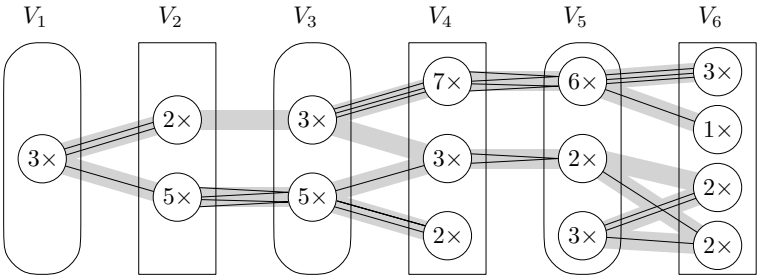


Figure 1: An example of $\mathcal{IV}$-matching.

$\mathcal{IV}$-matching is a subset of edges $M \subseteq E$ such that: Each vertex from an even layer V_{2k} is incident to exactly one vertex from $V_{2k-1} \cup V_{2k+1}$. Each vertex from the layer V_{2k+1} is either incident to exactly two vertices from V_{2k}, or exactly one vertex from V_{2k+2} (these two options are exclusive).

It implies that edges between layers V_{2k+1} and V_{2k+2} form a matching ($\mathcal{I}$-shapes) and edges between layers V_{2k} and V_{2k+1} form independent $\mathcal{V}$-shapes with centers in the layer V_{2k+1}.

As $\mathcal{IV}$-MATCHING we denote the decision problem of finding out whether there is an $\mathcal{IV}$-matching in a given graph.

PROBLEM:	$\mathcal{IV}$-MATCHING
Instance:	Graph G in the described format.
Question:	Is there an $\mathcal{IV}$-matching in G?

Note that for $\ell = 2$ the problem is just ordinary bipartite matching. For odd values of ℓ, the clusters of the layer V_ℓ can be matched in only one possible way, thus this odd case reduces to the case with $(\ell - 1)$ layers. First interesting case is $\ell = 4$ and we prove in the next section that this case is already $\mathcal{NP}$-complete.

2 $\mathcal{NP}$-completeness

We prove the $\mathcal{NP}$-completeness of $\mathcal{IV}$-MATCHING by reduction from the problem 3D-MATCHING.

Three-dimensional matching In the problem 3D-MATCHING we are given a hypergraph $H = (U, F)$. The set of vertices is split

into three equally sized partites X, Y and Z. Each edge consists of exactly one vertex from each partite thus $F \subseteq X \times Y \times Z$.

We call perfect matching a set of pairwise disjoint edges covering all vertices.

PROBLEM:	3D-MATCHING
Instance:	Hypergraph $H = (U, F)$ such that $F \subseteq X \times Y \times Z$.
Question:	Is there a perfect matching in G?

3D-MATCHING is well-known to be $\mathcal{NP}$-complete; it is actually the seventeenth problem in Karp's set of 21 $\mathcal{NP}$-complete problems [2].

Theorem 2.1. *The problem $\mathcal{IV}$-MATCHING is $\mathcal{NP}$-complete already for the case with $\ell = 4$.*

Proof. It is easy to see that the problem is in the $\mathcal{NP}$ class: The $\mathcal{IV}$-matching itself is a polynomial certificate and its corectness can be directly verified.

To begin with the reduction let $H = (U, F)$ be an instance of 3D-MATCHING with partites X, Y, Z and let us denote $n = |X| = |Y| = |Z|$ and $m = |F|$.

We construct the instance $G = (V, E)$ of $\mathcal{IV}$-MATCHING in this way: We put vertices from X and Y into the layer V_1 and vertices from Z into the layer V_4. Each vertex forms its own cluster of the size one. Then for each edge $e = \{x, y, z\}$ we add a cluster with two new vertices x_e, y_e into the layer V_2 and a cluster with one new vertex z_e into the layer V_3. We then add these four edges on clusters: $\langle\{x\}, \{x_e, y_e\}\rangle$, $\langle\{y\}, \{x_e, y_e\}\rangle$, $\langle\{x_e, y_e\}, \{z_e\}\rangle$ and $\langle\{z_e\}, \{z\}\rangle$. We call the vertices x_e, y_e and z_e *edge vertices* of e.

The key idea of this construction is that $\mathcal{V}$s in $\mathcal{IV}$-matching translate to edges *not* present in the perfect matching. You can see an example on the Figure 2.

The resulting instance of $\mathcal{IV}$-MATCHING has $3n + 3m$ vertices, $4m$ edges and it can be constructed directly in polynomial time. We shall now prove that there is an $\mathcal{IV}$-matching in G if and only if there is a perfect matching in the original hypergraph H.

$\Rightarrow$ Let M be an $\mathcal{IV}$-matching in G. For each hyperedge $e \in F$ observe that necessarily either all edge vertices x_e, y_e and z_e are matched with $\mathcal{I}$s or all of them are matched by one $\mathcal{V}$.

Let us put into our matching of H all edges $e \in F$ such that x_e, y_e and z_e are matched with $\mathcal{I}$s. Because M is an $\mathcal{IV}$-matching, every vertex of the original hypergraph is connected by $\mathcal{I}$ to exactly one edge vertex and we chose the corresponding edge to our matching

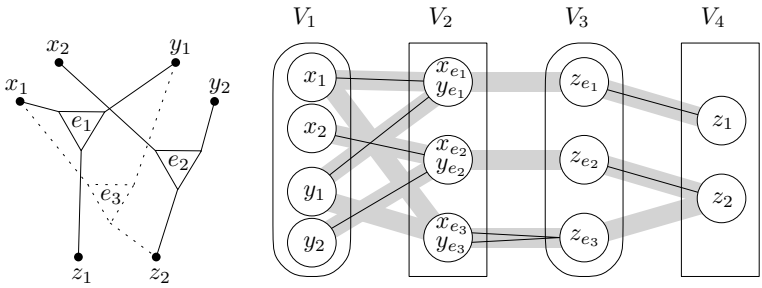


Figure 2: Instance of 3D-MATCHING with a perfect matching and equivalent $\mathcal{IV}$-MATCHING instance with an $\mathcal{IV}$-matching.

so all vertices of the hypergraph are covered. Because there are $3n$ vertices in V_1 and V_4 and $3m$ vertices in V_2 and V_3, the number of $\mathcal{V}$s used is $m - n$ and so we used n edges in our matching of the hypergraph. This proves that we constructed a perfect matching.

$\Leftarrow$ Let $N \subseteq F$ be a perfect matching in the hypergraph H. For each edge $e = \{x, y, z\}$ we connect by $\mathcal{I}$s the pairs of vertices $\{x, x_e\}$, $\{y, y_e\}$ and $\{z, z_e\}$. For each edge $e \notin N$ we cover the vertices x_e, y_e, z_e by a $\mathcal{V}$. We see that in this construction every vertex in V_2 and V_3 is covered by exactly one $\mathcal{I}$ or one $\mathcal{V}$.

Because matching N covers all vertices in H, every vertex in V_1 and V_4 is covered by at least one $\mathcal{I}$. Because we put $3n$ $\mathcal{I}$s into the graph, every vertex is covered by exactly one $\mathcal{I}$. This implies that this way we obtained a correct $\mathcal{IV}$-matching. □

3 Elimination of cut-vertices

In this section we prove that it is sufficient to study only 2-vertex-connected instances of the problem $\mathcal{IV}$-MATCHING. Firstly, we prove the following technical lemma and then the theorem itself.

Lemma 3.1. *Let $G = (C, E)$ be a graph on clusters where all clusters except for one have fixed size. If there is an $\mathcal{IV}$-matching in G, there is only one suitable size of the last cluster and can be computed in linear time (with respect to the compact input representation).*

Proof. Let us fix some $\mathcal{IV}$-matching in G and let us denote by $n_i, n_i^{\mathcal{I}}, n_i^{\mathcal{V}}$ the total number of vertices in V_i and the number of them matched

with $\mathcal{I}$s and $\mathcal{V}$s respectively. We assume that the cluster with unknown size is in the layer V_k.

Observe that for each i it holds $n^{\mathcal{I}}_{2i} = n^{\mathcal{I}}_{2i-1}$ and $n^{\mathcal{V}}_{2i} = 2n^{\mathcal{V}}_{2i+1}$. Because we know $n^{\mathcal{I}}_1 = n_1$ we can compute $n^{\mathcal{V}}_2 = n_2 - n^{\mathcal{I}}_2 = n_2 - n^{\mathcal{I}}_1$. Knowing $n^{\mathcal{V}}_2 = n^{\mathcal{V}}_3/2$ we have $n^{\mathcal{I}}_3 = n_3 - n^{\mathcal{V}}_3$. We repeat this procedure until we compute both $n^{\mathcal{I}}_{k-1}$ and $n^{\mathcal{V}}_{k-1}$.

Because we also know $n^{\mathcal{I}}_\ell = n_\ell$ or $n^{\mathcal{V}}_\ell = n_\ell$ depending on the parity of ℓ we can use a similar process to compute $n^{\mathcal{I}}_{k+1}$ and $n^{\mathcal{V}}_{k+1}$.

This computation running in linear time gave us $n^{\mathcal{I}}_{k-1}, n^{\mathcal{V}}_{k-1}, n^{\mathcal{I}}_{k+1}$ and $n^{\mathcal{V}}_{k+1}$. From this we also have the size of V_k and therefore there is only possible size for the cluster with unknown size so that V_k has the proper size. □

Theorem 3.2. *In linear time (with respect to the compact input representation) it is possible to decompose the instance of $\mathcal{IV}$-MATCHING into 2-vertex-connected parts such that the original instance has an $\mathcal{IV}$-matching if and only if each of the parts has an $\mathcal{IV}$-matching.*

Proof. The idea of this proof is to use Lemma 3.1 to split cut-vertices between components that share it. Instead of the term 2-vertex-connectivity we only say 2-connectivity in this proof.

Let us have an instance of $\mathcal{IV}$-MATCHING with the graph of clusters $G = (C, F)$. Let the tree $T = (V, E)$ be a decomposition of G into 2-connected components. Vertices V of the tree represent components of the decomposition and two components are connected by an edge if they share a cut-cluster.

This decomposition may be made in linear time by a standard technique involving depth-first search. Parts of our decomposition will be all the components of 2-connectivity, but sizes of cut-clusters in these components will be conveniently modified.

We find the proper sizes of cut-clusters and prove the correctness of our construction by induction on the number of vertices in the tree T.

In the case when T has only one component the whole graph is 2-connected and our decomposition consists of only one part – the whole graph G.

Let us assume that the theorem holds for instances with $k - 1$ components of 2-connectivity and that G has k components. Let L be a leaf-component in T and $c \in L$ the cut-cluster by which it is connected to the rest of G.

If we consider L to be a separate $\mathcal{IV}$-MATCHING instance, then Lemma 3.1 tells us that there is only one possible number of vertices

to be used in an $\mathcal{IV}$-matching of L if there exists any. Let this number be x; the lemma also tells us that it can be computed in linear time.

Let us denote by L' the component L where the size of the cluster c is set to x and by G' the graph G without $L \setminus \{c\}$ and with x vertices removed from the cluster c.

If there is an $\mathcal{IV}$-matching both in G' and L', then there is an $\mathcal{IV}$-matching in G because we can simply take the union of these matchings. If there is an $\mathcal{IV}$-matching in G, then the existence of $\mathcal{IV}$-matchings in G' and L' is implied by our use of Lemma 3.1.

By induction we can now decompose G' into $k-1$ proper parts and our decomposition is finished. The construction can be made in linear time because all our subtasks are done in linear time and ordering of the components in "leaf-order" can also be made in linear time. □

Acknowledgements

We would like to thank Pavel Klavík for introducing the problem and a fruitful discussion. Moreover we note that part of the research was conducted during the summer REU program 2014 at DIMACS, Rutgers University.

References

[1] Jiří Fiala, Pavel Klavík, Jan Kratochvíl and Roman Nedela *Algorithmic Aspects of Regular Graph Covers with Applications to Planar Graphs*, Lecture Notes in Computer Science, Automata, Languages, and Programming, pages 489–501, 2014.

[2] Richard M. Karp *Reducibility among Combinatorial Problems*, Complexity of Computer Computations, The IBM Research Symposia Series, pages 85–103, 1972.

Kakeya Problem for Binary Strings

Dušan Knop, Peter Korcsok, Karel Král

Abstract

For two binary strings $x, y \in \{0,1\}^n$ we define a line to be the set $L_{x,y} = \{x^i \oplus y \mid i = 0, \ldots, n-1\}$ where $\oplus$ stands for xor operation and x^i means the i-th rotation of x. We are interested in the smallest size of a set S such that for all x there exists an y such that $L_{x,y} \subseteq S$.

1 Introduction

The original meaning of a Kakeya set is a set $S \subseteq \mathbb{R}^n$ that is compact and contains a unit length segment in every direction. The conjecture is that such a set must have Hausdorff dimension equal to n. For more information see Bourgain [1] or references in Dvir [2].

Lately the analogy of this question was asked for finite fields. Let $\mathbb{F}$ stand for a finite field with q elements. We call a set $S \subseteq \mathbb{F}^n$ a Kakeya set if it contains a line in every direction. That is for every $x \in \mathbb{F}^n$ there is a point $y \in \mathbb{F}^n$ such that $L_{x,y} = \{y + a \cdot x \mid a \in \mathbb{F}\} \subseteq S$. Dvir [2] shows that the size of every Kakeya set is at least $C_n \cdot q^n$ for a constant C_n depending only on n and $q = |\mathbb{F}|$.

2 Problem Definition

Koucký [3] asks for the least size of a Kakeya set $S \subseteq \{0,1\}^n$ containing a line for every direction. We denote a line of a direction x as $L_{x,y} = \left\{y \oplus x^i \mid i \in \{0, \ldots, n-1\}\right\}$ where x^i is the i-th rotation of the binary word x and $\oplus$ stands for the xor operation. More formally:

Question 2.1. *What is the minimal size of a set $S \subseteq \{0,1\}^n$ such that for every $x \in \{0,1\}^n$ there is an $y \in \{0,1\}^n$ such that $L_{x,y} \subseteq S$.*

3 Observations

Observation 3.1. *For all $x, y \in \{0,1\}^n$ we have $\sum_{j=1}^{n} (y \oplus x^i)_j \equiv \sum_{i=1}^{n} (y_i + x_i) \pmod 2$.*

Proof. Let J_0 be the set of indexes where y is zero and let J_1 be the set of indexes where y is one. The proof follows by a direct computation $\sum_{j=1}^{n}(y \oplus x^i)_j \equiv \sum_{j\in J_0} x^i_j + \sum_{j\in J_1}(1 + x^i_j) \equiv \sum_{j\in J_0} x_{i+j} + \sum_{j\in J_1} x_{i+j} + \sum_{j\in J_1} 1 \equiv \sum_{i=1}^{n} x_i + |J_1| \equiv \sum_{i=1}^{n}(y_i + x_i) \pmod 2$.

□

Corollary 3.2. *There is a Kakeya set of size 2^{n-1}.*

Proof. Let $S = \{z \mid \sum z_i \equiv 0 \pmod 2\}$. For every x there is a y with the right parity of ones.

□

Observation 3.3. *For all $b, x \in \{0,1\}^n$ we have $\{y \mid L_{x,y} \ni b\} = L_{x,b}$.*

Proof. We have $x^i \oplus y = b$ iff $y = x^i \oplus b$.

□

Observation 3.4. *For every $b, x \in \{0,1\}^n$ we have that*

$$|\{y \mid L_{x,y} \ni b\}| \leq n.$$

Proof. $\forall i \exists! y\colon y \oplus x^i = b$ else we would have $y \oplus x^i = b = y' \oplus x^i$ and thus $y = y'$.

□

Corollary 3.5. *There is a Kakeya set of size $2^{n-1}(1 - 1/n)$.*

Proof. For any given x there are 2^{n-1} points y with the right parity of digits 1. When we want to remove a point $b \in S$ by Observation 3.4 we have at most n points y with $b \in L_{x,y}$, so we forbid using at most n such points. Doing this $\lfloor 2^{n-1}/n \rfloor$ times gives us the estimate.

□

Observation 3.6. *For every prime and $x \in \{0,1\}^n \setminus \{0^n, 1^n\}$ and for all $0 \leq i < j < n$ we have $x^i \neq x^j$.*

Corollary 3.7. *For every prime p we have $\frac{2^p-2}{p}$ integer.*

Observation 3.8. *$L_{x,y} = L_{\overline{x},\overline{y}}$.*

Observation 3.9. *Let $\mathcal{S}_n$ be the family of all Kakeya sets on $\{0,1\}^n$. For every $x \in \{0,1\}^n$ and $S \in \mathcal{S}_n$ we have $x \oplus S \in \mathcal{S}_n$ and $S \cup \{x\} \in \mathcal{S}_n$. For any $S \in \mathcal{S}_n$ we have $\{s^i \mid s \in S\} \in \mathcal{S}_n$ for any fixed i.*

Proof. For a given direction $d \in \{0,1\}^n$ we choose $y' = x \oplus y$.

By the definition a superset of a Kakeya set is a Kakeya set.

By the definition and rotating the string x.

□

Observation 3.10. *Intersection of lines of directions $x, \overline{x}$ needs not to be big.*

Proof. When we have $x^i \oplus y = \overline{x}^k \oplus \overline{y}$ and $x^j \oplus y = \overline{x}^l \oplus \overline{y}$ by xoring both together we get $\overline{x}^k \oplus x^i = x^j \oplus \overline{x}^l$.

□

Lemma 3.11. *When given a Kakeya set S we have another Kakeya set S' of the same or lesser size containing just vertices with the same parity of the number of digits* 1*.*

Proof. All points in one line have the same parity of digits 1.

For every $v \in S$ of odd parity (or even the same way) we negate the last bit of y that contributes some line into the odd part of S. This does not influence even lines and does not add more points than there were in the odd parity.

□

Question 3.12. *Is $L_{x,y} = \left\{x^i \oplus x \oplus y\right\}$ a better definition of a line?*

Observation 3.13. *For all Kakeya sets S we have an y such that all neighbors of y are in the Kakeya set $N(y) \subseteq S$.*

Proof. By the first definition with $x = 0000001$ we have y such that $L_{x,y} \subseteq S$ and the y is what we want.

□

Theorem 3.14. *The size of a Kakeya set is at least $|S| \geq 2^{\frac{n-1}{2}}$.*

Proof. Both definitions give us the same sizes of Kakeya sets. We choose the second definition of line $L_{x,y} = \left\{x^i \oplus x \oplus y \mid i = 0, \ldots, n-1\right\}$.

We know that we have to use $y \in S$ otherwise the first point $x \oplus x \oplus y$ is not in S.

For every x we look at the second point of the line. We would like to say that if x, z coincide on the second point we have either $z = x$ or

$z = \neg x$. We write the system of equations over $\mathbb{Z}_2$.

$$\begin{aligned} x_1 + x_2 &= z_1 + z_2 \\ x_2 + x_3 &= z_2 + z_3 \\ x_3 + x_4 &= z_3 + z_4 \\ &\dots \\ x_{n-1} + x_1 &= z_{n-1} + z_1 \end{aligned}$$

Thus we have to use at least k different ys where we have $|S| \geq \frac{2^{n-1}-1}{k} + 2$ and $k \leq |S|$ so $|S| \geq 2^{\frac{n-1}{2}} + 1$ holds.

□

Observation 3.15. *Let K_n be the size of the smallest Kakeya set. We have $K_n \leq K_{2n+\ell}$ for any $\ell \geq 0$.*

Proof. For an x of length n we choose $x0^\ell x$ and trim the first n bits from its y and n bits from the bigger Kakeya set.

□

n	1	2	3	4	5	6
K_n	1	2	3	6	9	14

Table 1: Computer experiments gave these least sizes of Kakeya sets for given n.

Using Observation 3.13 we can define a prototype of a Kakeya set in such a way that the y in Observation 3.13 is equal to zero and we can not get this set from another Kakeya set by any operation listed in Observation 3.9.

There is just one prototype of Kakeya set with $n = 4$ and it is

$$\{0001, 0010, 0100, 1000, 0111, 1101\}.$$

There are just two prototypes of Kakeya sets with $n = 5$ and those are

$$\{00001, 00010, 00100, 01000, 10000, 10011, 01011, 10101, 11111\}$$

and

$$\{00001, 00010, 00100, 01000, 10000, 01101, 00111, 11001, 11111\}.$$

Conjecture 3.16 (π-conjecture)**.** *We use the second definition of line. For every $n > 7$ prime, for every $x \in \{0,1\}^n$ such that $x \neq 0^n$ and $x \neq 1^n$ and for every $y \in \{0,1\}^n$ we have that*

$$|\bigcup_{i=0}^{n-1} L_{x^i,y}| = \binom{n}{2} + 1$$

and $\forall i, j \in \{0, \ldots, n-1\}, i \neq j \colon |L_{x^i y} \cap L_{x^j y}| = 2$.

References

[1] J. Bourgain *Harmonic analysis and combinatorics: How much may they contribute to each other?* IMU/Amer. Math. Soc., pages 13–32, 2000.

[2] Zeev Dvir *On the size of Kakeya sets in finite fields,* arXiv:0803.2336 [math.CO].

[3] Michal Koucký *personal communication*, 2014.

$L(2,1)$ - Labeling On Interval Graphs

Veronika Steffanová

1 Introduction

Interval graphs are intersection graphs of a family of intervals of real numbers. $L(p,q)$-labeling of a graph G is a mapping $l\colon V_G \to X$ where $X \subset \mathbb{Z}$ such that $|l(u) - l(v)| \geq p$ whenever the vertices u and v are connected by an edge and $|l(u) - l(v)| \geq q$ whenever there exists some vertex w such that both u and v are neighbours of w. Finally, span of graph G is the smallest number k such that there exists $L(p,q)$-labeling of G using $X = \{0, \ldots, k\}$. We found a formula for the span of $L(2,1)$-labeling for the class of interval graphs and its connection to the chromatic number of the graph and the maximum degree.

The problem of finding the minimal span was firstly announced by Jerrold R. Griggs and Roger K. Yeh in [2]. They presented the first estimate for general graphs $\lambda(G) \leq \Delta^2 + 2\Delta$, but gave a conjecture $\lambda(G) \leq \Delta^2$, where Δ is the maximum degree of G. They prove the conjecture for 2-regular graphs. They also showed that this problem is NP-complete for general graphs.

Their estimate was later improved in [3] $\lambda(G) \leq \Delta^2 + \Delta$ They also gave exact values of the span for trees: $\lambda(G) = \Delta + 1$ or $\Delta + 2$.

We studied the intersection graphs, which are subclass of the chordal graphs. Following estimate for the span of chordal graphs was given by Sakai in [1] and it is the only one, which was known for general interval graphs, too: $\lambda(G) \leq (\Delta + 3)^2/4$. This estimate proves the conjecture for chordal graphs.

By the same author the exact estimate for unit interval graphs was proved: $2\chi(G) - 2 \leq \lambda(G) \leq 2\chi$, where χ denote chromatic number of the graph G. Note that chordal graphs are perfect, so the chromatic number equals to the clique number.

2 Our results

Theorem 2.1. *There exists an algorithm which gives us correct $L(2,1)$-labeling of a given interval graph. It runs in polynomial time.*

The algorithm is based on the clique path of the graph and greedy algorithm going from the most left to the most right clique.

Theorem 2.2. *Let have an interval graph $G(V, E)$ of chromatic number χ and maximum degree Δ. Then*

$$\Lambda \leq 2\chi + \Delta - 2.$$

The estimate is derived from the greedy algorithm considering coloring of each clique separately.

Lemma 2.3. *There exist graphs with span $\lambda = 2\chi + \Delta - 2$ arbitrary large.*

The example which proves the lemma is a path of connected stars. The problem is the example has $\chi = 2$ and we were unable to find an example with higher chromatic number.

References

[1] D. Sakai *Labelling Chordal Graphs: Distance Two Condition* SIAM J. Discret. Math., pages 133–140, 1994.

[2] J. R. Griggs and R. K. Yeh *Labelling Graphs with a Condition at Distance 2* SIAM J. Discret. Math., pages 586–595, 1992.

[3] G. J. Chang and D. Kuo *The $L(2,1)$-Labeling Problem on Graphs* SIAM J. Discret. Math., pages 309–316, 1996.