Spring School 2018

Tomáš Masařík, Veronika Slívová, Jakub Svoboda, Jakub Tětek (eds.)

Preface

Spring school on Combinatorics has been a traditional meeting organized for over 30 years for faculty and students participating in the Combinatorial Seminar at Faculty of Mathematics and Physics of the Charles University. It is internationally known and regularly visited by students, postdocs and teachers from our cooperating institutions in the DIMATIA network. As it has been the case for several years, this Spring School is supported by Computer Science Institute (IÚUK) of Charles University, the Department of Applied Mathematics (KAM) and by some of our grants (SVV, UNCE). This year we are glad we can also acknowledge generous support by the RSJ Foundation.

The Spring Schools are entirely organized and arranged by our students (mostly undergraduates). The topics of talks are selected by supervisors from the Department of Applied Mathematics (KAM) and Computer Science Institute (IÚUK) of Charles University as well as from other participating institutions. In contrast, the talks themselves are almost exclusively given by students, both undergraduate and graduate. This leads to a unique atmosphere of the meeting which helps the students in further studies and their scientific orientation.

This year the Spring School is organized in Jáchymov, a historic small town in Krušné hory (Ore mountains) with a great variety of possibilities for outdoor activities.

Ondřej Pangrác, Robert Šámal, Martin Tancer

KAM CSI IÚUK ITI RSJ

Contents

Series on Secure Multi-Party Computation: Secret Sharing (Pavel Dvořák) 7
Series on Secure Multi-Party Computation: Multi-Party Computation - Semi-Honest Adversary (Veronika Slívová) 9
Series on Secure Multi-Party Computation: Malicious Adversary (Karel Král) 12
Series on Secure Multi-Party Communication: On Communication Complexity of Secure Computation (Pavel Hubáček) 15
Series on Gray Codes: Introduction to Gray Codes (Petr Gregor) 16
Series on Gray Codes: Optimal Quasi-Gray Codes: Does the Alphabet Matter? (Ondřej Mička) 17
Series on Gray Codes: Gray Codes and Symmetric Chains (Miloš Chromý) 19
A Simple, Space-Efficient, Streaming Algorithm for Matchings in Low Arboricity Graphs (Kateřina Altmanová) 21
Approximating Minimum-area Rectangular and Convex Containers for Packing Convex Polygons (Mikhail Beliayeu) 23
The Lovász Local Lemma and Variable Strength Covering Arrays (Václav Blažej) 24
The Chromatic Number of the Plane is at least 5 (Tomáš Gavenčiak) 25
Fermat's Last Theorem and Catalan's Conjecture in Arithmetics with Weak Exponentiation (Petr Glivický) 26
Machine Theory of Mind (Karel Ha) 27
Graph Classes of Linear Mim-Width 1 (Lars Jaffke) 28
Counterexamples to Jaeger's Circular Flow Conjecture (Anna Kompišová) 30
Parameterized Algorithm for Eternal Vertex Cover (Matyáš Křišťan) 32
Edge Colouring of Cubic Graphs (Robert Lukoťka) 34
Homomorphism reconfiguration via homotopy (Tomáš Masařík) 35
Series on Gray Codes: Optimal Quasi-Gray Codes: Does the Alphabet Matter? (Ondřej Mička) 38
Enumeration and Maximum Number of Minimal Connected Vertex Covers in Graphs (Jana Novotná) 40
Optimality of the Johnson-Lindenstrauss Lemma (Michal Opler) 42
Connected Vertex Cover is Polynomial-Time Solvable For $sP_1 + P_5$-Free Graphs (Giacomo Paesani) 43
Pseudo-Triangulations, Rigidity and Motion Planning (Sophie Rehberg) 44
Approximation Schemes for 0-1 Knapsack (Jakub Svoboda) 50
A Simple Algorithm for Approximating the Text-To-Pattern Hamming Distance (Jakub Tětek) 52
Minimal Asymmetric Graphs (Pablo Oviedo Timoneda) 53
A Simple Population Protocol for Fast Robust Approximate Majority (Martin Töpfer) . . . 54
Towards a Polynomial Kernel for Directed Feedback Vertex Set (Tomáš Toufar) 55
The Firefighting on Trees and Cayley Graphs (Lukáš Vik) 57

Schedule of talks

	9:00 – 10:30	10:30 – 12:00	12:00
Saturday *room A* *chair: Milan Hladík*	Series MPC: Semi-Honest Adversary Veronika Slívová 9:00 – 10:30 9	Optimality of the Johnson-Lindenstrauss Lemma Michal Opler 10:30 – 12:00 42	Lunch 12:00 – 13:00
Saturday *room B* *chair: Pavel Dvořák*		Series MPC: Malicious Adversary Karel Král 10:30 – 12:00 12	Lunch 12:00 – 13:00
Sunday *chair: Pavel Hubáček*	Minimal Asymmetric Graphs Pablo Oviedo Timoneda 9:00 – 10:30 53	Pseudo-Triangulations, Rigidity and Motion Planning Sophie Rehberg 10:30 – 12:00 44	Lunch 12:00 – 13:00
Monday *chair: Miloš Chromý*	Homomorphism Reconfiguration via Homotopy Tomáš Masařík 9:00 – 10:30 35	A Simple Algorithm for Approximating the Text-To-Pattern Hamming Distance Jakub Tětek 10:30 – 12:00 52	Lunch 12:00 – 13:00
Tuesday *room A* *chair: Jiří Fiala*	Series GC: Introduction to Gray Codes Petr Gregor 9:00 – 10:30 16	Series GC: Does the Alphabet Matter? Ondřej Mička 10:30 – 12:00 38	Lunch 12:00 – 13:00
Tuesday *room B* *chair: Karel Král*		Fermat's Last Theorem and Catalan's Conjecture in Arithmetics with Weak Exponentiation Petr Glivický 10:30 – 12:00 26	Lunch 12:00 – 13:00
Wednesday	Trip Day 9:00 –		
Thursday *room A* *chair: Tomáš Gavenčiak*	Approximation Schemes for 0-1 Knapsack Jakub Svoboda 9:00 – 10:30 50	A Simple Population Protocol for Fast Robust Approximate Majority Martin Töpfer 10:30 – 12:00 54	Lunch 12:00 – 13:00
Thursday *room B* *chair: Veronika Slívová*	Parametrized Algorithm for Eternal Vertex Cover Matyáš Křišťan 9:00 – 10:30 32		Lunch 12:00 – 13:00
Friday *chair: Ondřej Mička*	A Simple, Space-Efficient, Streaming Algorithm for Matchings in Low Arboricity Graphs Kateřina Altmanová 9:00 – 10:30 21	Approximating Minimum-area Rectangular and Convex Containers for Packing Convex Polygons Mikhail Beliayeu 10:30 – 12:00 23	Lunch 12:00 – 13:00

Friday *chair: Martin Tancer*	Arri-val – 17:00	Dinner 17:00 – 18:00	Graph Classes of Linear Mim-Width 1 Lars Jaffke 18:00 – 19:30 28	Series MPC: Secrete Sharing Pavel Dvořák 19:30 – 21:00 7
Saturday *room A* *chair: Tomáš Toufar*		Dinner 18:00 – 19:00	Series MPC: On Computation Complexity of Secure Computation Pavel Hubáček 19:00 – 20:30 15	
Saturday *room B*		Dinner 18:00 – 19:00		
Sunday *chair: Michal Opler*		Dinner 18:00 – 19:00	Towards a Polynomial Kernel for Directed Feedback Vertex Set Tomáš Toufar 19:00 – 20:30 55	
Monday *chair: Tomáš Masařík*		Dinner 18:00 – 19:00	Connected Vertex Cover is Polynomial-Time Solvable for $sP_1 + P_5$-Free Graphs Giacomo Paesani 19:00 – 20:30 43	
Tuesday *room A* *chair: Petr Gregor*		Dinner 18:00 – 19:00	Series GC: Gray Codes and Symmetric Chains Miloš Chromý 19:00 – 20:30 19	
Tuesday *room B* *chair: Tomáš Masařík*		Dinner 18:00 – 19:00	Enumeration and Maximum Number of Minimal Connected Vertex Covers in Graphs Jana Novotná 19:00 – 20:30 40	
Wednesday *chair: Tereza Klimošová*	Trip Day – 18:00	Dinner 18:00 – 19:00	Counterexamples to Jaeger's Circular Flow Conjecture Anna Kompišová 19:00 – 20:30 30	Edge Colouring of Cubic Graphs Robert Lukoťka 20:30 – 21:00 34
Thursday *room A* *chair: Martin Töpfer*		Dinner 18:00 – 19:00	The Lovász Local Lemma and Variable Strength Covering Arrays Václav Blažej 19:00 – 20:30 24	The Chromatic Number of the Plane is at least 5 Tomáš Gavenčiak 20:30 – 21:00 25
Thursday *room B* *chair: Jana Novotná*		Dinner 18:00 – 19:00	The Firefighting on Trees and Cayley Graphs Lukáš Vik 19:00 – 20:30 57	

Series Talks

Pavel Dvořák

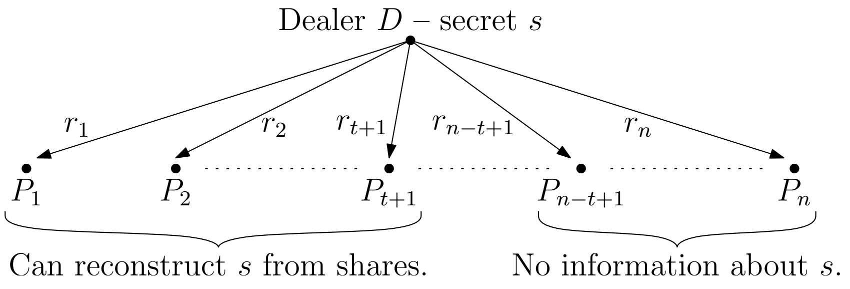
koblich@iuuk.mff.cuni.cz

Secret Sharing

as part of a serie Series on Secure Multi-Party Computation

Introduction

Informally, a $(t+1)$-out-of-n secret sharing scheme takes as input a secret s and outputs n shares, with the property that it is possible to efficiently reconstruct s from every subset of $t+1$ shares, but every subset of t or less shares reveals nothing about the secret s.



Reed-Solomon Code and Shamir's Secret Sharing Scheme

- We fix a field $\mathbb{F}$.

Definition 1 $[n,k,d]$-code *over $\mathbb{F}$ is a set $C \subseteq \mathbb{F}^n$ s.t.*

1. $|C| = |\mathbb{F}|^k$.
2. *Hamming distance of every two codewords in C is at least d.*

Reed-Solomon Code	
Parameters:	Distinct $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$ such that $|\mathbb{F}| > n$.
Message:	$(m_0, \ldots, m_t) \in \mathbb{F}^t$.
Codeword:	$C(m) = \Big(p_m(\alpha_1), \ldots, p_m(\alpha_n)\Big)$,
	where $p_m(x) = m_0 + m_1 x + \cdots + m_t x^t$.

Theorem 2 *The Reed-Solomon code is a $[n, t+1, n-t]$-code over $\mathbb{F}$. Moreover, there exists an efficient decoding algorithm that corrects up to $\frac{n-t-1}{2}$ errors.*

Shamir's Secret Sharing Scheme	
Dealer's Input:	A secret $s \in \mathbb{F}$.
Common Input:	Distinct, nonzero $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$.
Dealer Computes:	$m_0 = s$.
	$m_1, \ldots, m_t$ are random elements of $\mathbb{F}$.
	$q(x) = m_0 + m_1 x + \cdots + m_t x^t$.
Sharing:	Party P_i gets $q(\alpha_i)$.
Reconstruction:	$(t+1)$ parties have together $(t+1)$ points of polynomial q, thus they can compute $q(0) = s$.
Secrecy:	If at most t of polynomial points are fixed, then each value of the absolute term has the same probability.

What if Some Parties or Dealer are Dishonest?

- Let $t < n/3$, thus Reed-Solomon code can correct up to $\frac{3t+1-t-1}{2} = t$ errors.
- If there is a honest dealer and at most t dishonest parties, then honest parties can reveal the secret due to the Reed-Solomon code.
- Dealer have to embed the polynomial $q(x)$ (from Shamir's Secret Sharing Scheme) into a bivariate polynomial $S(x, y)$ such that honest parties can detect dishonest parties and dealer.

Verifiable Secret Sharing Scheme	
Dealer's Input:	A polynomial $q(x) \in \mathbb{F}[x]$ of degree at most t ($q(0)$ is the secret).
Common Input:	Distinct, nonzero $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$.
Dealer Computes:	$S(x, y)$ is a random polynomial, such that $S(0, z) = q(z)$.
Sharing:	The party P_i gets polynomials $f_i(x) = S(x, \alpha_i)$ and $g_i(y) = S(\alpha_i, y)$.
Consistency:	The party P_i sends $f_i(\alpha_j)$ and $g_i(\alpha_j)$ to the party P_j.
	The party P_j checks if $f_j(\alpha_i) = g_i(\alpha_j)$ $(= S(\alpha_i, \alpha_j))$ and $f_i(\alpha_j) = g_j(\alpha_i)$.
	If not, the party P_j makes a complaint about P_i to the dealer.
Resolve Complaints:	Parties with the dealer resolve complaints and eliminate dishonest parties.
Output:	If there is enough honest parties, then each party P_i sends $f_i(0) = q(\alpha_i)$.
	They reconstruct the secret from the shares due to the Reed-Solomon code.

Veronika Slívová
slivova@iuuk.mff.cuni.cz

Multi-Party Computation - Semi-Honest Adversary
as part of a serie Series on Secure Multi-Party Computation

Introduction

In the setting of secure multi-party computation n parties with (private) inputs want to securely compute some function of their inputs in the presence of adversary. Where securely computation means that the following properties are satisfied:

1. **privacy** – any party learns nothing from the protocol other than the output,
2. **correctness** – the output is distributed according to the prescribed functionality,
3. **independence of inputs** – parties cannot choose their inputs in dependence on other's inputs,
4. **fairness and guaranteed output delivery** – all parties receive the output.

There exist many different settings in which secure multi-party computation is studied:

1. according to adversary strategy
   (a) **semi-honest adversary** – adversary follows the prescribed protocol, but tries to learn more than they should by inspecting the protocol transcript,
   (b) **malicious adversary** – adversary may follow an arbitrary strategy,
2. according to adversary computational power
   (a) **computational setting** – the adversary is limited to polynomial time,
   (b) **information-theoretic setting** – the adversary is unbounded,
3. according to the corrupted parties
   (a) **static** – corrupted parties are fixed before the execution of protocol begins,
   (b) **adaptive** – adversary may adaptively choose to corrupt a party during the execution of the protocol,
4. according to the achieved security
   (a) **perfect security** – there is zero probability of cheating – the result in the real world is the same as in the ideal world,
   (b) **statistical security** – the result in the real world is statistically close to the result in the ideal world.

We denote the number of parties by n and the number of corrupted parties by t.

Main result and basic definitions

Theorem 1 *Consider a synchronous network with pairwise private channels and a broadcast channel. Then:*

1. *Semi-honest: For every n-ary functionality f, there exists a protocol for computing f with perfect security in the presence of static semi-honest adversary controlling up to $t < n/2$ parties,*

2. *Malicious: For every n-ary functionality f, there exists a protocol for computing f in the presence of a static malicious adversary controlling up to $t < n/3$ parties.*

The communication complexity of the protocol is $O(poly(n)|C|)$, where C is an arithmetic circuit computing f, and the round complexity is linear in the depth of the circuit C.

Definition 2 (view) *The* view *of the i-th party P_i during an execution of a protocol π on inputs $\vec{x}$, denoted by $VIEW_i^\pi(\vec{x})$ is defined to be $(x_i, r_i, m_{i_1}, \dots, m_{i_k})$, where x_i is P_i's private input, r_i is its internal coin tosses, and m_{i_j} is the j-th message that was received by P_i in the protocol execution.*

Definition 3 (t-privacy of n-party protocols – deterministic functionalities)) *Let $f\colon (0,1^*)^n \to (0,1^*)^n$ be a deterministic n-ary functionality and let π be a protocol. We say that π is t-*private *for f if for every $\vec{x} \in (0,1^*)^n$, where $|x_1| = \ldots = |x_n|$,*

$$OUTPUT^\pi(x_1, \dots, x_n) = f(x_1, \dots, x_n)$$

and there exists a probabilistic polynomial-time algorithm S such that for every $I \subset [n]$ (set of corrupted parties) of cardinality at most t and every $\vec{x} \in (0,1^*)^n$ where $|x_1| = \ldots = |x_n|$ it holds that:

$$\{S(I, \vec{x}_I, f_I(\vec{x}))\} = \{VIEW_I^\pi(\vec{x})\}$$

Protocol for t-Private Computation in the F_{mult}-Hybrid Model

- **Inputs:** Each party P_i has an input $x_i \in \mathbb{F}$

- **Auxiliary input:** Each party P_i has an arithmetic circuit C over the field $\mathbb{F}$ computing the functionality f. The parties aslo have a description of $\mathbb{F}$ and distinct nonzero values $\alpha_1, \dots, \alpha_n \in \mathbb{F}$.

- **The protocol**

  1. **Input-sharing stage** Each party P_i chooses a polynomial $q_i(x)$ uniformly from the set of all degree t polynomials with constant term x_i. For every j: P_i sends $q_i(\alpha_j)$ to P_j and records $q_1(\alpha_i), \dots, q_n(\alpha_i)$.

  2. **Circuit emulation stage** Let $G_1, \dots, G_\ell$ be a predetermined topological ordering of gates. For $k = 1, \dots, \ell$, let β_i^k (and γ_i^k) be the share(s) of input wire(s) held by P_i, the parties works as follows:

     (a) G_k is addition gate – P_i defines its share of output wire $\delta_i^k = \beta_i^k + \gamma_i^k$.

     (b) G_k is multiplication by constant c – P_i defines its share of output wire $\delta_i^k = c\beta_i^k$

     (c) G_k is multiplication – P_i sends β_i^k, γ_i^k to the ideal functionality F_{mult} and receives back the share of the output wire δ_i^k.

3. **Output reconstruction stage** Let $o_1, \ldots, o_n$ be output wires, where P_i's output is o_i. Let $\beta_1^k, \ldots, \beta_n^k$ be the shares that parties hold for wire o_k. Then each P_i sends P_k the share β_i^k. Upon receiving all shares, P_i computes $g_k(x)$ and define its output to be $g_k(0)$.

t-Privately Computing F_{mult}

- **Inputs:** Each party P_i holds values β_i, γ_i, such that $\beta_1, \ldots \beta_n$ as well as $\gamma_1, \ldots \gamma_n$ determines a polynomial of degree t.
- **The protocol**
  1. Each party locally computes $s_i = \beta_i \gamma_i$
  2. **Randomize:** each party P_i sends λ (empty string) to F_{rand}^{2t} and receives σ_i
  3. **Reduce degree:** each party P_i sends $(s_i + \sigma_i)$ to F_{reduce}^{deg} and receives δ_i.
  4. P_i outputs δ_i.

Privately Computing F_{rand}^{2t}

- **Inputs:** Parties do not have inputs
- **The protocol**
  1. Each P_i chooses a random polynomial $q_i(x) \in_R \mathcal{P}^{0,2t}$. Then for every $j \in \{1, \ldots, n\}$ it sends $s_{i,j} = q_i(\alpha_j)$ to party P_j.
  2. Each P_i receives $s_{1,i}, \ldots, s_{n,i}$ and computes $\delta_i = \sum_{j=1 \ldots n} s_{j,i}$.
  3. Party P_i outputs δ_i.

Bibliography

[1] Gilad Asharov, Yehuda Lindell *A Full Proof of the BGW Protocol for Perfectly Secure Multiparty Computation* `https://link.springer.com/article/10.1007%2Fs00145-015-9214-4`

Karel Král

kralka@iuuk.mff.cuni.cz

Malicious Adversary

as part of a serie Series on Secure Multi-Party Computation

Malicious Adversary

Definition 1 (Corruption-aware parallel VSS - F^n_{VSS}) *F^n_{VSS} receives a set of indices $I \subseteq [n]$ and works as follows:*

1. *F^n_{VSS} receives an input polynomial $q_j(x)$ from every* honest *P_j ($j \notin I$).*
2. *F^n_{VSS} sends the (ideal model) adversary the corrupted parties' shares $\{q_j(\alpha_i)\}_{j \notin I}$ for every $i \in I$, based on the honest parties' polynomials.*
3. *F^n_{VSS} receives from the (ideal model) adversary an input polynomial $q_i(x)$ for every $i \in I$.*
4. *F^n_{VSS} sends the shares $(q_1(\alpha_j), \dots, q_n(\alpha_j))$ to every party P_j ($j = 1, \dots, n$). If $\deg(q_i(x)) > t$ then $\perp$ is sent in place of $q_i(\alpha_j)$.*

Definition 2 (Functionality F^A_{mat} for matrix multiplication, with $A \in \mathbb{F}^{n \times m}$) *The F^A_{mat} functionality receives as input a set of indices $I \subseteq [n]$ and works as follows:*

1. *F^A_{mat} receives the inputs of the honest parties $\{g_j(x)\}_{j \notin I}$; if a polynomial $g_j(x)$ is not received or its degree is greater than t, then F^A_{mat} resets $g_j(x) = 0$.*
2. *F^A_{mat} sends shares $\{g_j(\alpha_i)\}_{j \notin I; i \in I}$ to the (ideal) adversary.*
3. *F^A_{mat} receives the corrupted parties' polynomials $\{g_i(x)\}_{i \in I}$ from the (ideal) adversary; if a polynomial $g_i(x)$ is not received or its degree is greater than t, then F^A_{mat} resets $g_i(x) = 0$.*
4. *F^A_{mat} computes $\vec{Y}(x) = (Y_1(x), \dots, Y_m(x)) = (g_1(x), \dots, g_n(x))A$.*
5. (a) *For every $j \notin I$, functionality F^A_{mat} sends party P_j the entire length-m vector $\vec{y} = \vec{Y}(0)$, together with P_j's shares $(g_1(\alpha_j), \dots, g_n(\alpha_j))$ on the input polynomials.*
   (b) *In addition, functionality F^A_{mat} sends the (ideal) adversary its output: the vector of polynomials $\vec{Y}(x)$ and the corrupted parties' outputs ($\vec{y}$ together with $(g_1(\alpha_i), \dots, g_n(\alpha_i))$, for every $i \in I$).*

Protocol 6.5 (Securely computing F^A_{mat} in the F_{VSS}-hybrid model)

- **Inputs:** Each party P_i holds a polynomial $g_i(x)$.
- **Common input:** A field description $\mathbb{F}, n$ distinct non-zero elements $\alpha_1, \dots, \alpha_n \in \mathbb{F}$, and a matrix $A \in \mathbb{F}^{n \times m}$.
- **Aiding ideal functionality initialization:** Upon invocation, the trusted party computing the corruption-aware parallel VSS functionality F^n_{VSS} is given the set of corrupted parties I.

- **The protocol:**

  1. Each party P_i checks that its input polynomial is of degree-t; if not, it resets $g_i(x) = 0$. It then sends its polynomial $g_i(x)$ to F^n_{VSS} as its private input.
  2. Each party P_i receives the values $g_1(\alpha_i), \ldots, g_n(\alpha_i)$ as output from F^n_{VSS}. If any value equals $\perp$, then P_i replaces it with 0.
  3. Denote $\vec{x}^i = (g_1(\alpha_i), \ldots, g_n(\alpha_i))$. Then, each party P_i locally computes $\vec{y}^i = \vec{x}^i A$ (equivalently, for every $k = 1, \ldots, m$, each P_i computes $Y_k(\alpha_i) = \sum_{\ell=1}^{n} g_\ell(\alpha_i) a_{\ell,k}$ where $(a_{1,k}, \ldots, a_{n,k})^T$ is the kth column of A, and stores $\vec{y}^i = (Y_1(\alpha_i), \ldots, Y_m(\alpha_i)))$.
  4. Each party P_i sends $\vec{y}^i$ to every P_j $(1 \le j \le n)$.
  5. For every $j = 1, \ldots, n$, denote the vector received by P_i from P_j by $\hat{\vec{Y}}(\alpha_j) = \hat{Y}_1(\alpha_j), \ldots, \hat{Y}_m(\alpha_j)$. (If any value is missing, it replaces it with 0. We stress that different parties may hold different vectors if a party is corrupted.) Each P_i works as follows:

     For every $k = 1, \ldots, m$, party P_i locally runs the Reed-Solomon decoding procedure (with $d = 2t + 1$) on the possibly corrupted codeword $(\hat{Y}_k(\alpha_1), \ldots, \hat{Y}_k(\alpha_n))$ to get the codeword $(Y_k(\alpha_1), \ldots, Y_k(\alpha_n))$. It then reconstructs the polynomial $Y_k(x)$ and computes $y_k = Y_k(0)$.

- **Output:** P_i outputs $(y_1, \ldots, y_m)$ as well as the shares $g_1(\alpha_i), \ldots, g_n(\alpha_i)$.

Protocol t-Secure Computation in the (F_{VSS}, F_{mult})-Hybrid Model

- **Inputs:** Each party P_i has an input $x_i \in \mathbb{F}$.
- **Common input:** Each party P_i holds an arithmetic circuit C over a field $\mathbb{F}$ of size greater than n, such that for every $\vec{x} \in \mathbb{F}^n$ it holds that $C(\vec{x}) = f(\vec{x})$ where $f\colon \mathbb{F}^n \to \mathbb{F}^n$. The parties also hold a description of $\mathbb{F}$ and distinct non-zero values $\alpha_1, \ldots, \alpha_n$ in $\mathbb{F}$.
- **Aiding ideal functionality initialization:** Upon invocation, the trusted party computing the (fictiously corruption-aware) functionality F_{VSS} and the corruption-aware functionality F_{mult} receive the set of corrupted parties I.
- **The protocol:**

  1. **The input sharing stage:**
     (a) Each party P_i chooses a polynomial $q_i(x)$ uniformly at random from the set of degree-t polynomials with constant-term x_i. Then, P_i invokes the F_{VSS} functionality as dealer, using $q_i(x)$ as its input.
     (b) Each party P_i records the values $q_1(\alpha_i), \ldots, q_n(\alpha_i)$ that it received from the F_{VSS} functionality invocations. If the output from F_{VSS} is $\perp$ for any of these values, P_i replaces the value with 0.
  2. **The circuit emulation stage:** Let $G_1, \ldots, G_\ell$ be a predetermined topological ordering of the gates of the circuit. For $k = 1, \ldots, \ell$ the parties work as follows:
     - Case 1 – G_k is an addition gate: Let β_i^k and γ_i^k be the shares of input wires held by party P_i. Then, P_i defines its share of the output wire to be $\delta_i^k = \beta_i^k + \gamma_i^k$.

- Case 2 – G_k is a multiplication-by-a-constant gate with constant c: Let β_i^k be the share of the input wire held by party P_i. Then P_i defines its share of the output wire to be $\delta_i^k = c \cdot \beta_i^k$.
- Case 3 – G_k is a multiplication gate: Let β_i^k and γ_i^k be the shares of input wires held by party P_i. Then P_i sends (β_i^k, γ_i^k) to the ideal functionality F_{mult} and receives back a value δ_i^k. Party P_i defines its share of the output wire to be δ_i^k.

3. **The output reconstruction stage:**

   (a) Let $o_1, \ldots, o_n$ be the output wires, where party P_i's output is the value on wire o_i. For every $i = 1, \ldots, n$ denote by $\beta_1^i, \ldots, \beta_n^i$ the shares that the parties hold for wire o_i. Then each P_j sends P_i the share β_j^i.

   (b) Upon receiving all shares, P_i runs the Reed-Solomon decoding procedure on the possible corrupted codeword $(\beta_1^i, \ldots, \beta_n^i)$ to obtain a codeword $(\overline{\beta}_1^i, \ldots, \overline{\beta}_n^i)$. Then P_i computes $\text{reconstruct}_{\vec{\alpha}}(\overline{\beta}_1^i, \ldots, \overline{\beta}_n^i)$ and obtains a polynomial $g_i(x)$. Finally P_i then defines its output to be $g_i(0)$.

Pavel Hubáček

hubacek@iuuk.mff.cuni.cz

On Communication Complexity of Secure Computation *as part of a serie* Series on Secure Multi-Party Communication

I will present some long-standing open problems about communication complexity of secure computation and explain a lower bound on communication complexity for protocols securely computing functions with a long output.

Petr Gregor

gregor@ktiml.mff.cuni.cz

Introduction to Gray Codes *as part of a serie* Series on Gray Codes

Introduction

A Gray code for some combinatorial class (of bitstrings, permutations, partitions, trees, etc) is a (cyclic) enumeration of all its elements so that every two consecutive elements differ in some specific way. It corresponds to a Hamilton cycle in a suitably defined graph. We give a brief introduction to the area of binary Gray codes. Then we present a short proof of the middle levels theorem obtained in a joint work with Torsten Mütze and Jerri Nummenpalo [1].

Bibliography

[1] P. Gregor, T. Mütze, and J. Nummenpalo, *A short proof of the middle levels theorem*, Discrete Analysis, accepted. Preprint available at arXiv:1710.08249.

Ondřej Mička

`micka@ktiml.mff.cuni.cz`

Presented paper by Diptarka Chakraborty, Debarati Das, Michal Koucky, and Nitin Saurabh

Optimal Quasi-Gray Codes: Does the Alphabet Matter?
as part of a serie Series on Gray Codes

(`https://arxiv.org/abs/1712.01834v1`)

Introduction

A quasi-Gray code is a sequence of binary of strings over some alphabet (e.g. elements of $\mathbb{Z}_m^n$) such that consecutive strings differ in only small number of position. Such codes are tightly connected to the counters – a seemingly simple data structure that represents integer with increment operation. When designing counters we try to minimize the number of cells we need to read in order to increment, number of cells we need to write and we want to maximize the length of the counter (how many different integers it can represent).

This article presents a novel approach to the construction of quasi-Gray codes (or counters) based on algebraic tools. This leads to space-optimal quasi-Gray codes for odd-sized alphabet with logarithmic read complexity and write complexity 2. For binary alphabet, authors achieved similar complexity, while the code misses only linearly many strings.

In this talk, we will focus on the first part of the article. We will show tools used for composing and decomposing counters and we will use them to construct the quasi-Gray code for binary alphabet mentioned above.

Preliminaries

Definition 1 (Counter) *A counter of length l over domain $\mathcal{D}$ is a cyclic sequence $C = (w_1, w_2, \ldots, w_l)$ of distinct elements from $\mathcal{D}$. We define successor function $next(C, w_i)$ such that $next(C, w_i) = w_{i+1}$ for every $i \in [l-1]$ and $next(C, w_l) = w_1$. If $l = |\mathcal{D}|$, we say that counter is* space-optimal.

Definition 2 (Gray code) *A Gray code of length l over domain $\mathcal{D} = \mathcal{D}_1 \times \cdots \times \mathcal{D}_n$ is counter of length l over $\mathcal{D}$ such that w_i and $next(C, w_i)$ differ in exactly one coordinate. By allowing consecutive tuples to differ in at most c coordinates for some fixed constant $c \geq 1$, we get a c-*Gray code *or quasi-Gray code if c is not specified.*

Definition 3 (Decision Assignment Tree) *Let $\mathcal{D}^n$ be a domain and $\langle x_1, \ldots, x_n \rangle$ be a variables for an elements of that domain.* Decision Assignment Tree *(DAT) is a rooted $|\mathcal{D}|$-ary tree such that each internal node is labeled by one of the x_i variables and each outgoing edges are labeled with a distinct element of $\mathcal{D}$ (these represent read operations on the input). Each leaf node is labeled by a set of assignment instructions assigning some fixed values to a subset of variables.*

The read complexity *of DAT T,* READ(T)*, is the maximum length of any path from root to the leaf. The* write complexity, WRITE(T)*, is the maximum number of assignments in any leaf node.*

Length	Read complexity	Write complexity
2^n	$n-1$	1
2^{n-1}	$\log n + 4$	4
$2^n - 2^{n-t}$	$\log n + t + 3$	2

An example of few known constructions of (quasi-)Gray codes over $\mathbb{Z}_2^n$.

Now, we can state the main results of the article (we will prove only the second one in this talk).

Theorem 4 *Let $m \in \mathbb{N}$ be odd and $n \in \mathbb{N}$ be sufficiently large. Then there is a space-optimal quasi-Gray code C over $\mathbb{Z}_m^n$ such that next(C, w) can be implemented by DAT with read complexity at most $4 \log_m n$ and write complexity 2.*

Theorem 5 *Let $n \in \mathbb{N}$ be sufficiently large. Then there is a quasi-Gray code C of length at least $2^n - 20n$ over $\mathbb{Z}_2^n$ such that next(C, w) can be implemented by DAT with read complexity at most $6 + \log n$ and write complexity 2.*

Chinese remainder theorem for counters

The first tool is based on the well-known Chinese remainder theorem. Using this tool, we can compose efficient counters for small domains into the counter for larger domain, while keeping the complexities reasonably small.

Theorem 6 (Chinese Remainder Theorem) *Let $n_1, \dots, n_k$ are pairwise co-prime integers, $N = \prod_{i=1}^{k} n_i$ and $a_1, \dots, a_k$ are arbitrary integers. Then there exists exactly one integer $0 \le x < N$ such that $x \equiv a_i \pmod{n}_i$ for every $i = 1, \dots, k$. In other words, there is a bijection (isomorphism in fact) between $\mathbb{Z}_N$ and $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2} \times \cdots \times \mathbb{Z}_{n_k}$.*

Theorem 7 (Chinese Remainder Theorem for Counters) *Let $r \in \mathbb{N}$ and C_i for $i \in [r]$ are counters of length l_i over $\mathcal{D}_i$ computed by DAT T_i. Moreover, let $l \geq r - 1$ and $l_2, \dots, l_r$ are pairwise co-prime. Then there exists a DAT T implementing counter C of length $\prod_{i=1}^{r} l_i$ over $\mathcal{D}_1 \times \cdots \times \mathcal{D}_r$. Furthermore $\mathrm{READ}(T) = n_1 + \max_{i=2}^{r}(\mathrm{READ}(T_i))$ and $\mathrm{WRITE}(T) = \mathrm{WRITE}(T_1) + \max_{i=2}^{r}(\mathrm{WRITE}(T_i))$, where n_1 number of variables in T_1.*

Permutations and decomposition of counters

This tool is based on the fact, that counters are basically nothing more than a cycles in permutations. By smart decomposition of the underlying permutation, we can design DAT with low complexity.

Lemma 8 *Let $\mathcal{D}$ be a domain, $\sigma_1, \dots, \sigma_k$ permutations on $\mathcal{D}$ such that $\sigma = \sigma_k \circ \sigma_{k-1} \circ \dots \sigma_1$ is cycle of length l and $T_1, \dots, T_k$ DATs implementing $\sigma_1, \dots, \sigma_k$. Let $\mathcal{D}'$ be domain and T' a DAT implementing a counter C' of length $k' \geq k$ over $\mathcal{D}'$.*

Then there exists a DAT T implementing a counter C of length $k'l$ over $\mathcal{D} \times \mathcal{D}'$ such that $\mathrm{READ}(T) = r' + \max_{i=1}^{k}(\mathrm{READ}(T_i))$ and $\mathrm{WRITE}(T) = \mathrm{WRITE}(T') + \max_{i=1}^{k}(\mathrm{WRITE}(T_i))$, where r' is number of variables in T'

Counters via linear transformation

Finally, we show the construction of a counter that gives the Theorem 5. The counter is defined through a repeated application of a particular linear mapping – represented by a matrix of full rank. By decomposing matrix into elementary matrices and using the lemma from previous section, we get the desired counter.

Lemma 9 *Every invertible matrix $A \in \mathbb{F}^{n \times n}$ can be written as product of at most $n^2 + 4(n-1)$ elementary matrices.*

Lemma 10 *Let $z \in \mathbb{Z}_q^n$ be a root of primitive polynomial and A be a matrix representing a mapping $x \mapsto z \cdot x$. Then $\mathbb{Z}_q^n = \{0, 1, z, z^2, \dots, z^{q^n-2}\}$ and A can be decomposed into at most $n + 4(n-1)$ elementary matrices.*

Miloš Chromý

chromy@ktiml.mff.cuni.cz

Presented paper by Petr Gregor, Sven Jäger, Torsten Mütze, Joe Sawada, Kaja Wille

Gray Codes and Symmetric Chains

as part of a serie Series on Gray Codes

(https://arxiv.org/abs/1802.06021)

Introduction

Generating Gray codes is a problem of constructing a cyclic listing of all bitstrings of a length n with hamming distance of each two following bitstrings is one. This code has widespread use, e.g. in circuit design and testing, signal processing and error correction etc.

In this talk we mention the middle $2l$ levels problem where we are looking for a Gray code of middle $2l$ levels of a hypercube Q_n. This problem is connected to a cycle factor of middle $2l$ levels of Q_n which can be constructed from symmetric chain decomposition(SCD) of a hypercube Q_n.

Apart from building Gray codes, SCD have many other interesting applications like construction of rotation-symmetric Venn diagrams for n sets, where n is a prime number, a Littlewood-Offord problem on sums of vectors, etc.

The talk will be mostly about a symmetric chain decomposition.

Notation

As *Gray code* we will understand a Hamiltonian cycle in some subset of n dimensional hypercube Q_n (vertices are bitstrings of a length n and two vertices are connected iff their Hamming distance is 1) induced by $2l$ middle levels for $l = 1, \ldots, n+1$.

A *cycle factor* is a collection of disjoint cycles which together visit all vertices of the graph. In particular, Hamilton cycle is a cycle factor consisting only of a single cycle. This cycle factor can be obtained from two edge-disjoint symmetric chain decompositions.

A *symmetric chain* in is a path $(x_k, x_{k+1}, \ldots, x_{n-k})$ in the n-cube where x_i is from level i for all $k \leq i \leq n-k$. A *symmetric chain decomposition (SCD)* is a partition of the vertices of Q_n into symmetric chains.

Middle $2l$ levels problem

Since introduction of binary reflected Gray code, there has been continued interest in developing Gray codes for bitstrings of length n that satisfy various additional constraints. One of the studied constraints is generating Gray code of middle two levels of a cube Q_n [3], for cube without vertices $00\ldots0$ and $11\ldots1$ (middle $2n$ levels) [1] and middle $2(n-1)$ levels [2].

Problem M *(middle $2l$ levels problem.) For any $n \geq 1$ and $1 \leq l \leq n+1$ construct a cyclic listing of all bitstrings of length $2n+1$ with Hamming weights in the interval $[n+1-l, n+l]$ by flipping a single bit in each step.*

Theorem 1. *For any $n \geq 1$, the subgraph of the $(2n+1)$-cube induced by the middle four levels has a Hamilton cycle.*

For general $2l$ levels a cycle factor can possibly be used as a starting point to Hamiltonian cycle.

Theorem 2. *For any $n \geq 1$ and $1 \leq l \leq n+1$, the subgraph of the $(2n+1)$-cube induced by the middle $2l$ levels has a cycle factor.*

Symmetric chain decomposition

A Symmetric chain decomposition can be used to find a cycle factor of middle $2l$ levels cube Q_n.

There is well-known construction of two edge disjoint SCDs in the n-cube for any $n \geq 1$. This raises question how many of edge disjoint SCDs we can generate for any n-cube. In this paper we have an answer for $n \geq 13$ and some special cases in following theorems.

Theorem 3. *For any even $n \geq 6$, the n-cube contains four pairwise edge-disjoint SCDs.*

Theorem 4. *For $n = 7$ and any odd $n \geq 13$, the n-cube contains four pairwise edge-disjoint SCDs.*

Theorem 5. *If Q_a and Q_b each contain k pairwise edge-disjoint SCDs, then Q_{a+b} contains k pairwise edge-disjoint SCDs.*

Bibliography

[1] M. El-Hashash and A. Hassan. On the Hamiltonicity of two subgraphs of the hypercube. In *Proceedings of the Thirty-second Southeastern International Conference on Combinatorics, Graph Theory and Computing (Baton Rouge, LA, 2001)*, volume 148, pages 7–32, 2001

[2] P. Gregor and R. Škrekovski. On generalized middle-level problem. *Inform. Sci.*, 180(12):2448–2457, 2010.

[3] T. Mütze and J. Nummenpalo. A constant-time algorithm for middle levels Gray codes. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 2238–2253, 2017.

Standalone Talks

Kateřina Altmanová

kacka@kam.mff.cuni.cz

Presented paper by Andrew McGregor, Sofya Vorotnikova

A Simple, Space-Efficient, Streaming Algorithm for Matchings in Low Arboricity Graphs

(https://people.cs.umass.edu/ mcgregor/papers/18-sosa.pdf)

Introduction

In this talk we are going to describe a simple single-pass data algorithm using $O(\varepsilon^{-2}\log n)$ space that returns a $(\alpha+2)(1+\varepsilon)$ approximation to the size of the maximum matching in a graph of an arboricity α.

Definitions & Notations

Let $\boldsymbol{G = (V, E)}$ be a graph.

Arboricity $\boldsymbol{\alpha}$ of a graph G is the lowest number of forests into which its edges can be partitioned.

Let $\boldsymbol{match(G)}$ be the maximum size of a matching in a graph G.

Let $\boldsymbol{E_\alpha}$ be the set of edges $uv \in G$ where the number of edges incident to u or v that appear in the stream after uv are both at most α.

Define $\boldsymbol{H}$ to be set of vertices with degree $\geq \alpha + 1$. And we refer to these as the **heavy vertices**.

For $u \in V$, let $\boldsymbol{B_u}$ be the set of the last $\alpha + 1$ edges incident to u that arrive in the stream.

An edge uv is **good** if $uv \in B_u \bigcap B_v$ and **wasted** if $uv \in B_u \oplus B_v$, i.e., the symmetric difference.

We define $\boldsymbol{w}$ = number of good edges with no end points in H, $\boldsymbol{x}$ = number of good edges with exactly one end point in H, $\boldsymbol{y}$ = number of good edges with two end points in H, $\boldsymbol{z}$ = number of wasted edges with two end points in H.

Let $\boldsymbol{G_t}$ be the graph defined by the stream prefix of leght t and let $\boldsymbol{E_\alpha^t}$ be the set of good edges with respect to this prefix.

Theorems & Lemmas & Corollaries

Theorem 1. $match(G) \leq |E_\alpha| \leq (\alpha+2)match(G)$

Corollary of Edmonds' Theorem. Recall that the Edmond's Theorem implies that if the weight of a fractional matching on any induced subgraph $G(U)$ is at most $(|U|-1)/2$, then the weight on the entire graph is at most $match(G)$.

Lemma 1. Let $\{y_e\}_{e\in E}$ be a fractional matching where the maximum weight is ε. It holds that $\sum_e y_e \leq (1+\varepsilon)match(G)$.

Theorem 2. With high probability, the algorithm outputs a $(1+\varepsilon)$ approximation of $max_t|E_\alpha^t|$.

Theorem 3. The size of the maximum matching of a graph with arboricity α can be $(\alpha+2)(1+\varepsilon)$-approximated with high probability using a single pass over the edges of G given $O(\varepsilon^{-2}\log n)$ space.

The Algorithm

The algorithm is a modification of an algorithm for estimating $|E_\alpha|$ designed by Cormode et al. [1].

```
Initialize $S \longleftarrow \emptyset$, $p = 1$, $estimate = 0$
for each edge $e = uv$ in the stream do
    With probability $p$ add $e$ to $S$ and initialize counters $c_e^u \longleftarrow 0$ and $c_e^v \longleftarrow 0$
    for each edge $e' \in S$, if $e'$ shares endpoint $w$ with $e$ do
        Increment $c_{e'}^w$,
        If $c_{e'}^w \geq \alpha$, remove $e'$ and corresponding counters from $S$
    if $|S| > 40\varepsilon^{-2} \log n$ then
        $p \longleftarrow p/2$
        Remove each edge in $S$ and corresponding counters with probability $1/2$
    $estimate \longleftarrow max(estimate, |S|/p)$
return $estimate$
```

Bibliography

[1] Graham Cormode, Hossein Jowhari, Morteza Monemizadeh, S. Muthukrishnan. *The Sparse Awekens: Streaming Algorithm for Matching Size Estimation in Sparse Graphs.* Algorithms - ESA 2017 - 25th Annual European Symposium, September 4-6, 2017, Proceedings, 2017.

Mikhail Beliayeu

mikhail.beliayeu@gmail.com

Presented paper by Helmut Alt, Mark de Berg, Christian Knauer

Approximating Minimum-area Rectangular and Convex Containers for Packing Convex Polygons

(http://jocg.org/index.php/jocg/article/view/289)

Introduction

We investigate the problem of finding a minimum-area container for the disjoint packing of a set of convex polygons by translations. In particular, we consider axis-parallel rectangles or arbitrary convex sets as containers. For both optimization problems which are NP-hard we develop efficient constant factor approximation algorithms.

Rectangular Containers

Definition 1 *Let P be a set of convex polygons, $height(p) :=$ difference between maximum and minimum y-coordinates of a polygon p, $0 < \alpha < 1$ and $h_{max} := \max_p height(p)$. We partition P into **height classes** $P_0, P_1, ...$ such that $p \in P_i \iff h_{i+1} < height(p) \leq h_i$, where $h_i = \alpha^i h_{max}$.*

Algorithm 1

1. Pack each height class P_i separately into a container B_i of height h_i.
2. Replace each nonempty container B_i by a collection of axis-aligned *mini-containers* that are not too wide. Pack all mini-containers into a single container B.

Lemma 2 *The area of the container B_i computed for P_i satisfies*

$$area(B_i) \leq 2/\alpha \cdot \sum_{p \in P_i} area(p) + 2h_i \cdot \max_{p \in P_i} width(p)$$

Theorem 3 *Let P be a set of polygons in the plane with n vertices in total. We can pack P in $O(n \log n)$ time into an axis-aligned rectangular container B such that $area(B) \leq 17.45 \cdot OPT$, where OPT is the minimum area of any axis-aligned rectangular container for P.*

Convex Containers

Theorem 4 *Let P be a set of convex polygons in the plane with n vertices in total. We can pack P in $O(n \log n)$ time into a convex polygon B such that $area(B) \leq 27 \cdot OPT$, where OPT is the minimum area of any convex container for P.*

Václav Blažej

vaclav.blazej@fit.cvut.cz

Presented paper by Lucia Moura, Sebastian Raaphorst and Brett Stevents

The Lovász Local Lemma and Variable Strength Covering Arrays

(https://www.sciencedirect.com/science/article/pii/S1571065318300465)

Introduction

Lovász Local Lemma is a probabilistic technique to get an existence proof in settings where events are only sparsely dependent. We will use this lemma to get an upper bound on variable strenght covering array (VCA). We will compare this bound with known bounds on several classes of hypergraphs.

Definition 1 (VCA & VCAN) *Let $H = (V, E)$ be a hypergraph and let $k = |V|$. A* variable strength covering array*, denoted $VCA(n; H, v)$, is an $n \times k$ array M filled from $\mathbb{Z}_v$ such that for $e = \{v_0, \ldots, v_{t-1}\} \in E$, the $n \times t$ subarray of columns indexed by e is* covered*, that is it has every possible t-tuple in $\mathbb{Z}_v$ as a row at least once. The* variable-strength covering array number*, written $VCAN(H, v)$, is the smallest n such that a $VCA(n; H, v)$ exists.*

Theorem 2 (Lovász local lemma - Symmetric Case) *Consider a finite set of events $\mathcal{A} = \{A_0, \ldots, A_{m-1}\}$ in a probability space Ω such that each event occurs with probability at most $p < 1$, and each event is independent of all but at most d of the other events. If $ep(d + 1) \leq 1$, where e is the base of the natural logarithm, then the probability that none of the events occur is nonzero.*

Main result

Theorem 3 *Let $H = (V, E)$ be a hypergraph with $rank(H) = t \geq 1$, and let d be an integer such that no edge of H intersects more than d other edges of H. Then, for any $v \geq 2$, we have:*

$$VCAN(H, v) \leq \left\lceil \frac{\ln(d+1) + t \ln v + 1}{\ln \frac{v^t}{v^t - 1}} \right\rceil \tag{1}$$

Definition 4 *A* design *is a pair $(X, \mathcal{A})$ such that the following properties are satisfied: X is a set of elements called* points*, and $\mathcal{A}$ is a collection (i.e., multiset) of nonempty subsets of X called* blocks*.*

Definition 5 *Let v, k, and λ be a positive integers such that $v > k \geq 2$. A t-(v, k, λ)* balanced incomplete block design (BIBD) *is a design $(X, \mathcal{A})$ such that the following properties are satisfied: $|X| = v$, each block contains exactly k points, each t-element subset of X is contained in exactly λ blocks.*

Sometimes we use notation (v, b, r, k, λ) to emphasize number of blocks r and in how many blocks each point occurs r. These two parameters depend only on v, k, λ.

Definition 6 *The* cyclic consecutive hupergraph *is $H_c^{k,t} = (V, E)$ with $V = \{0, \ldots, k-1\}$ and $E = \{\{i, i + 1 \bmod k, \ldots, i + t - 1 \bmod k\} : 0 \leq i \leq k - 1\}$*

Definition 7 *A* triangulation hypergraph of the sphere*, $T = (V, E)$ is a rank-3 hypergraph which corresponds to a planar graph all of whose faces are triangles; the rank-3 hyperedges are precisely the faces of the planar embedding.*

Tomáš Gavenčiak

gavento@kam.mff.cuni.cz

Presented paper by Aubrey D.N.J. de Grey

The Chromatic Number of the Plane is at least 5

(https://arxiv.org/pdf/1804.02385.pdf)

A short and visual report on an elegant improvement of the plane coloring problem (Hadwiger-Nelson problem). We will state the problem, recall the known simple upper and lower bounds, and sketch the main idea of a construction showing that the chromatic number of the plane is at least 5.

Petr Glivický

petrglivicky@gmail.com

Presented paper by P. Glivický, V. Kala

Fermat's Last Theorem and Catalan's Conjecture in Arithmetics with Weak Exponentiation

(https://arxiv.org/pdf/1602.03580.pdf)

Abstract

Wiles's proof of Fermat's Last Theorem (FLT) has stimulated a lively discussion on how much is actually needed for the proof. Despite the fact that the original proof uses set-theoretical assumptions unprovable in Zermelo-Fraenkel set theory with axiom of choice (ZFC) - namely, the existence of Grothendieck universes - it is widely believed that "certainly much less than ZFC is used in principle, probably nothing beyond Peano arithmetic, and perhaps much less than that." (McLarty)

In this talk, I will present a joint work with V. Kala. We studied (un)provabiliy of FLT and Catalan's conjecture in arithmetical theories with weak exponentiation, i.e. in theories in the language $L = (0, 1, +, x, exp, <)$ where the $(0, 1, +, x, <)$-fragment is usually very strong (often even the complete theory $Th(N)$ of natural numbers in that language) but the exponentiation satisfies only basic arithmetical properties and not much of induction. In such theories, Diophantine problems such as FLT or Catalan's conjecture, are formalized using the exponentiation exp instead of the exponentiation definable in the $(0, 1, +, x, <)$-fragment.

I will present a natural basic set of axioms Exp for exponentiation (consisting mostly of elementary identities) and show that the theory $T = Th(N) + Exp$ is strong enough to prove Catalan's conjecture, while FLT is still unprovable in T. This gives an interesting separation of strengths of the two famous Diophantine problems. Nevertheless, I show that by adding just one more axiom for exponentiation (the, so called, "coprimality" of exp) the theory becomes strong enough to prove FLT, i.e. FLT is provable in T+"coprimality". (Of course, in the proof of this, we use the Wiles's result too.)

Karel Ha

mathemage@gmail.com

Presented paper by Neil C. Rabinowitz, Frank Perbet, H. Francis Song, Chiyuan Zhang, S.M. Ali Eslami, Matthew Botvinick

Machine Theory of Mind

(https://arxiv.org/abs/1802.07740)

Abstract

Theory of mind (ToM; Premack & Woodruff, 1978) broadly refers to humans' ability to represent the mental states of others, including their desires, beliefs, and intentions. We propose to train a machine to build such models too. We design a Theory of Mind neural network – a ToMnet – which uses meta-learning to build models of the agents it encounters, from observations of their behaviour alone. Through this process, it acquires a strong prior model for agents' behaviour, as well as the ability to bootstrap to richer predictions about agents' characteristics and mental states using only a small number of behavioural observations. We apply the ToMnet to agents behaving in simple gridworld environments, showing that it learns to model random, algorithmic, and deep reinforcement learning agents from varied populations, and that it passes classic ToM tasks such as the "Sally-Anne" test (Wimmer & Perner, 1983; Baron-Cohen et al., 1985) of recognising that others can hold false beliefs about the world. We argue that this system – which autonomously learns how to model other agents in its world – is an important step forward for developing multi-agent AI systems, for building intermediating technology for machine-human interaction, and for advancing the progress on interpretable AI.

Lars Jaffke
lars.jaffke@uib.no

Graph Classes of Linear Mim-Width 1

Joint work with Jan Arne Telle.

Introduction

Maximum Induced Matching Width (or mim-width for short) is a width parameter for graphs that was introduced by Vatshelle in 2012 [3]. One of its distinguishing features is that many well-studied graph classes such as e.g. Interval, Permutation or Circular Arc graphs have constant mim-width [1]. In fact, the *linear* mim-width of Interval and Permutation graphs is just 1, whereas these classes contain graphs whose clique-width is proportional to the square-root of the number of their vertices. In this talk we discuss a recently observed characterization of graphs of linear mim-width 1 in terms of linear orders defined with respect to neighborhood containment; and invite to further investigation.

Preliminaries

For a positive integer n, we let $[n] = \{1, \ldots, n\}$. Let G be a graph. For two (disjoint) vertex sets $X, Y \subseteq V(G)$, we denote by $G[X, Y]$ the bipartite subgraph of G with bipartition (X, Y) such that for $x \in X$, $y \in Y$, x and y are adjacent in $G[X, Y]$ if and only if they are adjacent in G. For a vertex set $X \subseteq V(G)$, we denote by $\text{mim}_G(X)$ the maximum size of any induced matching in $G[X, V(G) \setminus X]$.

Definition 1 *Let G be an n-vertex graph and $\sigma\colon V(G) \to [n]$ a linear order on its vertices. Let $v_1, \ldots, v_n$ denote the vertices of G ordered according to σ. The* maximum induced matching width *of σ is defined as $\max_{i\in[n-1]} \text{mim}_G(\{v_1, \ldots, v_i\})$. The* linear mim-width *of G is defined as the minimum mim-width over all linear orders of $V(G)$.*

Definition 2 *Let G be an n-vertex graph and $\sigma\colon V(G) \to [n]$ a linear order and denote by $v_1, \ldots, v_n$ the vertices of G ordered according to σ. A linear order $\rho\colon V(G) \to [n]$ is called*

1. *a* left neighborhood nesting w.r.t. σ *if for all $i \in [n]$ and $x, y \in \{v_1, \ldots, v_i\}$ with $\rho(x) \le \rho(y)$,*
$$N(x) \cap \{v_{i+1}, \ldots, v_n\} \subseteq N(y) \cap \{v_{i+1}, \ldots, v_n\}, \text{ and}$$
2. *a* right neighborhood nesting w.r.t. σ *if for all $i \in [n]$ and $x, y, \in \{v_i, \ldots, v_n\}$ with $\rho(x) \le \rho(y)$,*
$$N(x) \cap \{v_1, \ldots, v_{i-1}\} \subseteq N(y) \cap \{v_1, \ldots, v_{i-1}\}.$$

If ρ is a left (right) neighborhood nesting w.r.t. σ then we also say that σ admits *the left (right) neighborhood nesting ρ.*

The following observation is made from the fact that a bipartite graph has maximum induced matching size 1 if and only if it is a bipartite chain graph.

Observation 3 *Let G be a graph and $\sigma\colon V(G) \to [n]$ a linear order.*

1. *There is a left neighborhood nesting w.r.t. σ if and only if there is a right neighborhood nesting w.r.t. σ.*
2. *The order σ has linear mim-width 1 if and only if σ admits a neighborhood nesting.*

Characterizations of LM-1 Graph Classes

Investigating graph classes of linear mim-width 1 through the lens of neighborhood nestings, we find the following characterization of INTERVAL graphs.

Lemma 4 *An n-vertex graph G is an* INTERVAL *graph if and only if there exists a linear order $\sigma\colon V(G) \to [n]$ that admits a neighborhood nesting ρ with $\sigma = \rho$.*

For PERMUTATION graphs, we have the following (partial) characterization, where for a linear order $\rho\colon V(G) \to [n]$, its *reverse* $\texttt{rev}(\rho) = \rho'$ is defined as $\rho'(x) = n - \rho(x) + 1$ for all $x \in V(G)$.

Proposition 5 *If an n-vertex graph G is a* PERMUTATION *graph then there exists a linear order $\sigma\colon V(G) \to [n]$ that admits a left neighborhood nesting ρ_L and a right neighborhood nesting ρ_R such that $\rho_R = \texttt{rev}(\rho_L)$.*

We expect the condition stated in the previous proposition to be a precise characterization for PERMUTATION graphs. We therefore pose the following conjecture.

Conjecture 6 *A graph G is a* PERMUTATION *graph* if and only if *there exists a linear order $\sigma\colon V(G) \to [n]$ that admits a left neighborhood nesting ρ_L and a right neighborhood nesting ρ_R such that $\rho_L = \texttt{rev}(\rho_R)$.*

On all graph classes that are known to have linear mim-width 1 the HAMILTONIAN CYCLE problem is solvable in polynomial time. Very recently, a superset of the authors showed that HAMILTONIAN CYCLE is NP-complete on graphs of linear mim-width 1 [2], implying that there are more graphs of linear mim-width 1. Neighborhood nestings might be a useful tool in further identifying natural (sub-) classes of graphs that have linear mim-width 1.

Open Question 7 *Can other relations (and operations) on neighborhood nestings lead to more characterizations of natural graph classes of linear mim-width 1?*

Bibliography

[1] Rémy Belmonte and Martin Vatshelle. *Graph Classes with Structured Neighborhoods and Algorithmic Applications. Theoretical Computer Science*, 2013.

[2] Lars Jaffke, O-joung Kwon, and Jan Arne Telle. *A Unified Polynomial-Time Algorithm for Feedback Vertex Set on Graphs of Bounded Mim-Width.* In *STACS 2018.*

[3] Martin Vatshelle. *New Width-Parameters of Graphs.* PhD Thesis, *University of Bergen, Norway,* 2012.

Anna Kompišová

kompisova@dcs.fmph.uniba.sk

Presented paper by Miaomiao Han, Jiaao Li, Yezhou Wu and Cun-Quan Zhang

Counterexamples to Jaeger's Circular Flow Conjecture

(https://www.sciencedirect.com/science/article/pii/S0095895618300029)

Introduction

In 1981, Jaeger proposed the following conjecture (Circular Flow Conjecture or Modulo Orientation Conjecture):

Conjecture 1 *Every $4p$-edge-connected graph admits a modulo $(2p+1)$-orientation*

Kochol also suggested a seemly weaker conjecture:

Conjecture 2 *Every $(4p+1)$-edge-connected graph admits a modulo $(2p+1)$-orientation.*

We will show, that these conjectures are generally not true. Jaeger's conjecture doesn't hold for $p \geq 3$ and Kochol's conjecture for $p \geq 5$.

Theorem 3 *For every integer $p \geq 3$, there exists a $4p$-edge-connected graph admitting no modulo $(2p+1)$-orientation.*

Theorem 4 *For every integer $p \geq 5$, there exists a $(4p+1)$-edge-connected graph admitting no modulo $(2p+1)$-orientation.*

Definitions

Let D be an orientation of edges of graph G, let $E^+_D(v)$ be the set of outgoing edges from v and $E^-_D(v)$ be the set of ingoing edges to v.

Definition 5 *Orientation D of graph G is called modulo k-orientation if for every $v \in V(G)$ holds:*

$$|E^+_D(v)| \equiv |E^-_D(v)| \pmod{k}$$

In the construction of counterexamples we will use simple operation, which makes from smaller graphs with no modulo $(2p+1)$-orientation the new graph with the same property. The operation is called 2-sum.

Definition 6 *Let H_1 and H_2 be two graphs with $u_1, v_1 \in V(H_1)$, $u_2, v_2 \in V(H_2)$, such that number of parallel edges between u_1 and v_1 is at least $(2p-1)$. Define $H = H_1 \oplus H_2$, the 2-sum of H_1 and H_2, to be the graph obtained from H_1 and H_2 by deleting $2p-1$ edges between u_1 and v_1 in H_1 and then identifying u_1 and u_2 to be a new vertex u, and identifying v_1 and v_2 to be a new vertex v.*

Construction for theorem 3

Construction starts with complete graph K_{4p} on $4p$ vertices which does not admit modulo $(2p+1)$-orientation. Let $p \geq 3$ be an integer and $\{v_1, v_2, \ldots v_{4p}\}$ be the vertex set of the complete graph K_{4p}.

Steps of construction are as follows:

1. Construct the graph G_1 from K_{4p} by adding an additional set T of edges such that $V(T) = \{v_1, v_2, \ldots, v_{3(p-1)}\}$ and each component of the edge-induced subgraph $G_1[T]$ is a triangle.

2. Construct a graph G_2 from G_1 by adding two new vertices z_1 and z_2, adding one edge z_1z_2, adding $(p-2)$ parallel edges connecting v_{4p} and z_j for $j = 1, 2$ and adding one edge v_iz_j for each $3p-2 \leq i \leq 4p-1$ and $j = 1, 2$.
3. Denote by C_{4p+1} the cycle of length $4p+1$ with $V(C) = \{c_i : i \in \mathbb{Z}_{4p+1}\}$ and $E(C_{4p+1}) = \{c_ic_{i+1} : i \in \mathbb{Z}_{4p+1}\}$. Let $W = (2p-1)C_{4p+1} \cdot K_1$ be the graph obtained from C_{4p+1} by replacing each edge c_ic_{i+1} with $2p-1$ parallel edges and then adding a center vertex joining each vertex c_i in the cycle.
4. For each c_i, c_{i+1} $(i \in \mathbb{Z}_{4p+1})$ in W and z_1, z_2 in copy of G_2, apply the 2-sum operation. Let M be the resulting graph.

In the proof we show that graphs G_1, G_2 and W admit no modulo $(2p+1)$-orientation. The validity of the theorem then comes from property of 2-sum operation and the fact, that M is $4p$-edge-connected graph.

Construction for theorem 4

The construction and the proof of theorem 4 are almost the same as for the theorem 3. The main difference is that $p \geq 5$ and instead of graph G_2 we are using another graph G_3 with no edge-cut of size $4p$ not separating z_1 and z_2.

Graph G_3 is constructed as follows:

1. Construction starts with complete graph K_{4p} with vertices $\{v_1, v_2, \ldots v_{4p}\}$.
2. Let $q = \lceil \frac{2p-1}{3} \rceil$. Construct graph G' from the complete graph K_{4p} by adding an additional set T' of edges such that $V(T') = \{v_1, v_2, \ldots, v_{3q}\}$ and each component of the edge-induced subgraph $G'_1[T]$ is a triangle.
3. Construct a graph G'_2 from G'_1 by adding two new vertices z'_1 and z'_2, adding one edge $z'_1z'_2$, adding $(3q-2p+2)$ parallel edges connecting v_{4p-1} and z'_j for $j = 1, 2$ and adding one edge $v_iz'_j$ for each $3q+1 \leq i \leq 4p-2$ and $j = 1, 2$.
4. Let G_2^1, G_2^2, G_2^3 be three copies of G'_2. Construct a graph G_3 from these three copies of G'_2 by identifying the corresponding z'_1 in G_2^1 and G_2^2 to be a new vertex y_1 identifying the corresponding z'_2 in G_2^2 and G_2^3 to be a new vertex y_2, and adding a triangle connecting the corresponding v_{v4}'s of G_2^1, G_2^2 and G_2^3. Relabel z'_2 in G_2^1 as x_1 and z'_1 in G_2^3 as x_2.

The last step is to apply the 2-sum operation for each c_i, c_{i+1} $(i \in \mathbb{Z}_{4p+1})$ in W and x_1, x_2 in copy of G_3. Let M' be the resulting graph.

Outline of the proof is basically the same as for the theorem 3.

Matyáš Křišťan
kristja6@fit.cvut.cz

Presented paper by Fedor V. Fomin, Serge Gaspers, Petr A. Golovach, Dieter Kratsch, Saket Saurabh

Parameterized Algorithm for Eternal Vertex Cover

(https://www.sciencedirect.com/science/article/pii/S0020019010001602)

Introduction

In this paper, we will talk about a combinatorial game played on a graph. The first player controls a set of guards, which he initially places on vertices of the graph. The second player repeatedly chooses one edge on which he attacks. The first player must defend against the attack by moving one of his guards along the attacked edge, otherwise he loses the game. During his turn, he can move each guard into a neighboring vertex. Note that if some configuration of guards is able to defend against any attack, the vertices on which the guards are placed create a vertex cover. Eternal Vertex Cover is a configuration of guards, which can defend the graph against any infinite sequence of attacks.

The main result is a parameterized algorithm for deciding, whether k guards can eternally defend the graph. We do that by showing that the problem admits a kernel of size $4^k(k+1)+2k$ and also show that the problem is fixed parameter tractable. We also present a proof of NP-hardness for the problem and a 2-approximation algorithm.

Parameterized complexity

Definition 1 *FPT (fixed parameter tractable) is a class of problems, which may be solved in time $f(k)n^{\mathcal{O}(1)}$, where n is the size of the input, k is some parameter of the input and f is some computable function.*

Definition 2 *Kernelization is transformation of instance of some problem into a smaller instance, for which the result is the same as for the original instance. The resulting smaller instance is called a kernel.*

Definitions

Definition 3 ***Eternal Vertex Cover*** *is a set of guards placed on vertices of the graph, which, given certain rules, can eternally defend G from attacks on any of its edges. The rules are as follows:*

- *At the beginning of each turn, only one edge will be attacked.*
- *At least one of the guards must move along the attacked edge.*
- *A guard can move from his vertex to any neighboring vertex.*
- *Any amount of guards can move in the same turn.*
- *The resulting configuration of guards must again be an Eternal Vertex Cover.*

Notation

$vc(G)$ denotes the size of the smallest vertex cover on G.

$evcn(G)$ dontes the size of the smallest Eternal Vertex Cover on G.

Theorems

Lemma 4 *For any graph G, $vc(G) \leq evcn(G) \leq 2vc(G)$.*

Proposition 5 *It is NP-hard to decide whether k guards can eternally protect all the edges of a graph.*

Lemma 6 *There is a 2-approximation algorithm for finding evcn(G) with running time $\mathcal{O}(\sqrt{n} \cdot m)$, where n is the number of vertices and m is the number of edges of the input graph.*

Lemma 7 *Deciding whether $(\mathcal{I}, \mathcal{J})$ is an e-edge in the configuration graph $\mathcal{G}$ can be done in time $n^{\mathcal{O}(1)}$.*

Lemma 8 *For a graph G, $evcn(G) \leq k \iff$ the configuration graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ for G and k is non-empty after recursively removing all the unsafe vertices of $\mathcal{G}$, and this property can be checked in time $\mathcal{O}(|\mathcal{E}| \cdot |\mathcal{V}|)$.*

Theorem 9 *Given a graph G on n vertices, it is possible to compute $evcn(G)$ in time $\mathcal{O}(64^n \cdot n^{\mathcal{O}(1)})$*

Lemma 10 *Let G be a graph on n vertices and m edges. Then $evcn(G) \leq k \iff evcn(G(S)) \leq k$, where $G(S)$ is the kernel for G and $G(S)$ can be constructed in time $\mathcal{O}(nm)$.*

Theorem 11 *Given a graph G on n vertices and m edges and integer k, we can obtain a kernel for Eternal Vertex Cover with at most $k' = 4^k(k+1) + 2k$ vertices in time $\mathcal{O}(nm)$.*

Theorem 12 *Let G be a graph on n vertices and m edges and k be a positive integer. It is possible to check whether $evcn(G) \leq k$ in time $\mathcal{O}(2^{\mathcal{O}(k^2)} + nm)$.*

Robert Lukoťka

lukotka@dcs.fmph.uniba.sk

Edge Colouring of Cubic Graphs

Abstract

We discuss selected ideas and algorithms that can be used to improve running times of the algorithms deciding the chromatic index of cubic graphs. We show two ways how to transform a 3-edge-coloring into improper 2-vertex coloring using balanced valuations [1, 2] or embeddings into orientable surfaces. Finally, we sketch how Hilbert's Nullstellensatz can be used to provide a certificate that a graph is not 3-edge-colourable.

Bibliography

[1] F. Jaeger: Balanced Valuations and Flows in Multigraphs, Proceedings of the American Mathematical Society Vol. 55, No. 1 (Feb., 1976), pp. 237-242.

[2] S. L. Hakimi: On the degrees of the vertices of a directed graph, J. Franklin Inst.279(1965), 290-308. MR31 4736.

[3] J. A. De Loera, J. Lee, P. N. Malkin, S. Margulies: Computing infeasibility certificates for combinatorial problems through Hilbert's Nullstellensatz, Journal of Symbolic Computation Volume 46, Issue 11, November 2011, Pages 1260-1283.

Tomáš Masařík

maso@kam.mff.cuni.cz

Presented paper by Marcin Wrochna

Homomorphism reconfiguration via homotopy

(https://arxiv.org/abs/1408.2812)

Abstract

The paper consider the following problem for a fixed graph H: given a graph G and two H-colorings of G, i.e. homomorphisms from G to H, can one be transformed (reconfigured) into the other by changing one color at a time, maintaining an H-coloring throughout. This is the same as finding a path in the $\mathrm{Hom}(G, H)$ complex. For $H = K_k$ this is the problem of finding paths between k-colorings, which was shown to be in P for $k \leq 3$ and PSPACE-complete otherwise by Cereceda et al. 2011. We generalize the positive side of this dichotomy by providing an algorithm that solves the problem in polynomial time for any H with no C_4 subgraph. This gives a large class of constraints for which finding solutions to the Constraint Satisfaction Problem is NP-complete, but finding paths in the solution space is P.

The algorithm uses a characterization of possible reconfiguration sequences (paths in $\mathrm{Hom}(G, H)$, whose main part is a purely topological condition described in algebraic terms of the fundamental groupoid of H seen as a topological space.

Figures

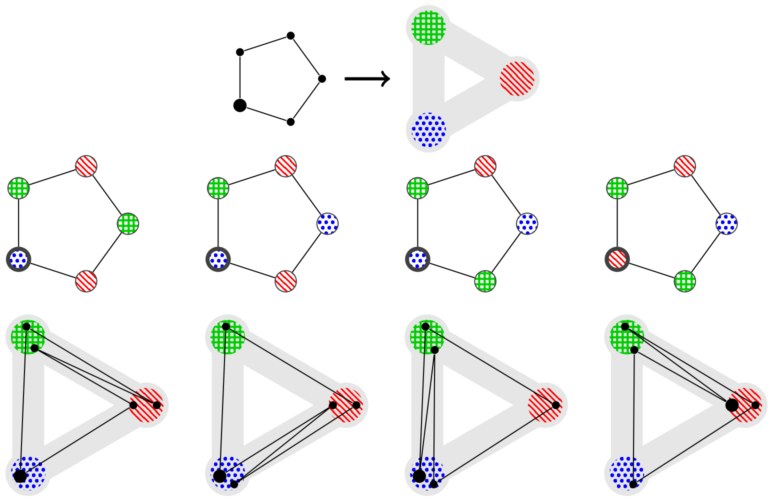


Figure 1: A sequence of 3-colorings of C_5 and the same sequence seen as a K_3-recoloring sequence of homomorphisms from C_5 to K_3 (a graph with three vertices: striped red, checkered green, dotted blue). One vertex of C_5 is thickened for clarity.

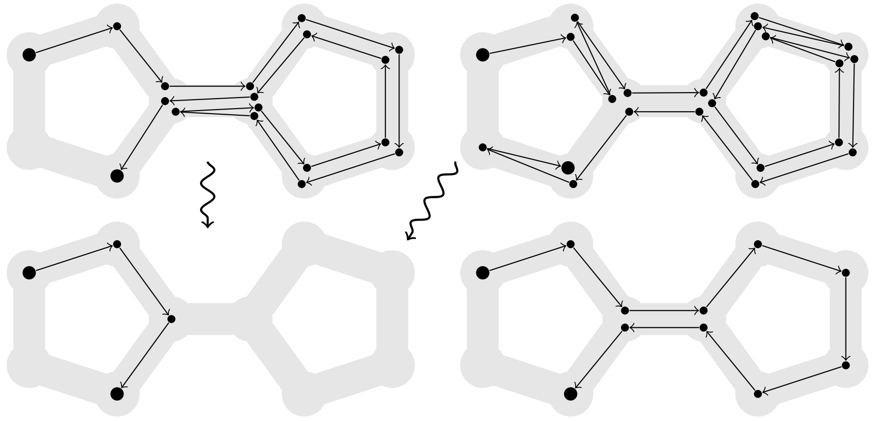


Figure 2: Examples of two walks (in a 'dumbbell' graph H on 10 vertices) which reduce to the same, bottom left one. The bottom right one is a different reduced walk; when its endpoints are fixed, it cannot be distorted as a curve to give any of the others.

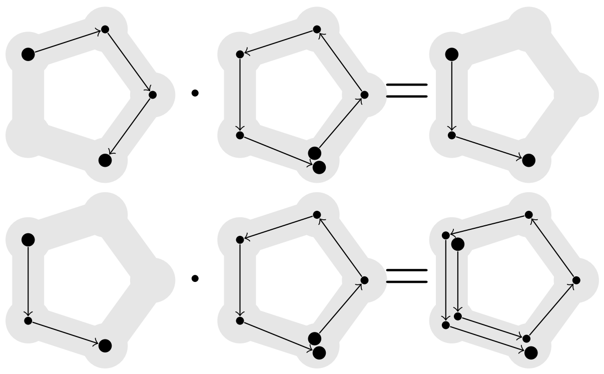


Figure 3: Examples of $\cdot$ multiplication in the fundamental groupoid of $H = C_5$.

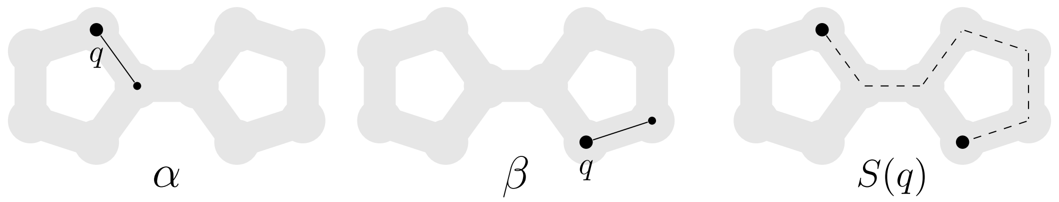


Figure 4: A realizable walk for $\alpha, \beta : K_2 \to H$ and q. Note that the shortest walk from $\alpha(q)$ to $\beta(q)$ (of length 3) is not realizable because of parity.

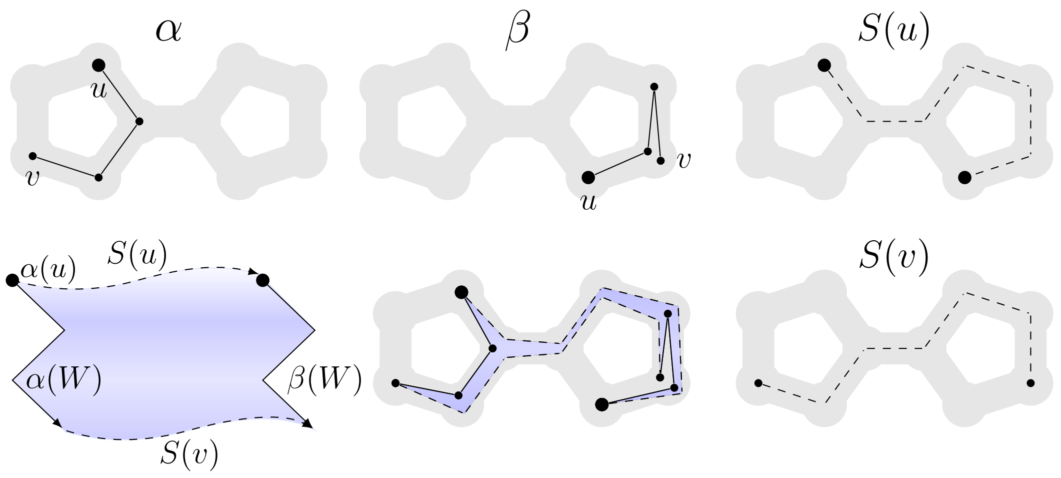


Figure 5: Intuitively, if α can be transformed to β by reconfiguration, then it can by a homotopy $\varphi : [0,1] \times [0,|W|] \to H$ such that $\varphi(0,\cdot) = \alpha(W)$ and $\varphi(1,\cdot) = \beta(W)$. Let $S(u) = \varphi(\cdot,0)$ and $S(v) = \varphi(\cdot,|W|)$. Since φ is a continuous mapping of a rectangle to H and since the boundary of the rectangle can be contracted to a point, the image of this boundary can also be contracted: $\overline{\alpha(W)}^{-1} \cdot \overline{S(u)} \cdot \overline{\beta(W)} \cdot \overline{S(v)}^{-1} = \varepsilon$.

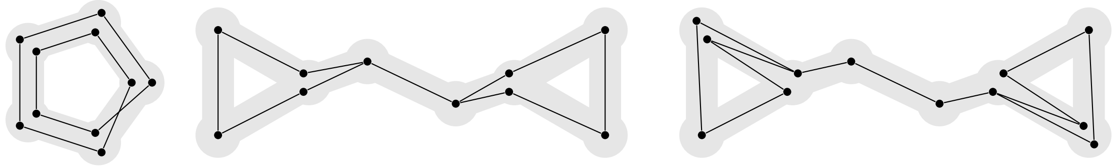


Figure 6: Left: walking along the 10 edges of the thin black graph gives a tight walk containing all vertices, so no reconfiguration step is possible. Middle: walking along one cycle, the bridge, the second cycle, and then back along the bridge, gives a tight closed walk containing all vertices, so no reconfiguration step is possible. Right: no closed walk is tight, but the 4 middle vertices are frozen.

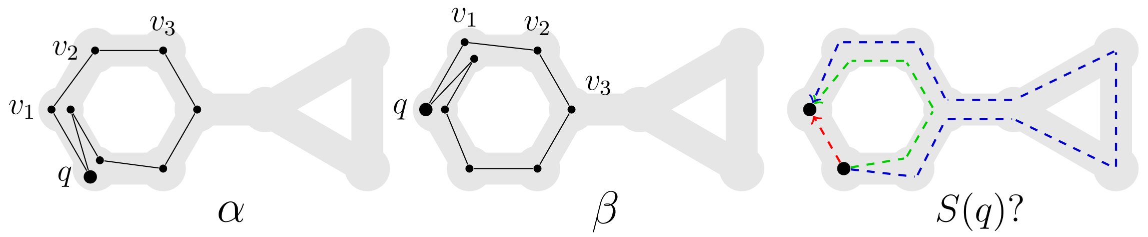


Figure 7: Two H-colorings α, β of an 8-cycle, where H is the gray graph on 9 vertices. Even though no vertex is frozen, α cannot be reconfigured to β. The red and green walks are not realizable for α, β, q because of parity. The blue walk has good parity, but is not topologically valid (imagine continuously deforming the 8-cycle by pulling q along the blue path—the cycle would necessarily end up stretched around the triangle).

Ondřej Mička
micka@ktiml.mff.cuni.cz

Presented paper by Diptarka Chakraborty, Debarati Das, Michal Koucky, and Nitin Saurabh

Optimal Quasi-Gray Codes: Does the Alphabet Matter?

as part of a serie Series on Gray Codes

(https://arxiv.org/abs/1712.01834v1)

Introduction

A quasi-Gray code is a sequence of binary of strings over some alphabet (e.g. elements of $\mathbb{Z}_m^n$) such that consecutive strings differ in only small number of position. Such codes are tightly connected to the counters – a seemingly simple data structure that represents integer with increment operation. When designing counters we try to minimize the number of cells we need to read in order to increment, number of cells we need to write and we want to maximize the length of the counter (how many different integers it can represent).

This article presents a novel approach to the construction of quasi-Gray codes (or counters) based on algebraic tools. This leads to space-optimal quasi-Gray codes for odd-sized alphabet with logarithmic read complexity and write complexity 2. For binary alphabet, authors achieved similar complexity, while the code misses only linearly many strings.

In this talk, we will focus on the first part of the article. We will show tools used for composing and decomposing counters and we will use them to construct the quasi-Gray code for binary alphabet mentioned above.

Preliminaries

Definition 1 (Counter) *A counter of length l over domain $\mathcal{D}$ is a cyclic sequence $C = (w_1, w_2, \dots, w_l)$ of distinct elements from $\mathcal{D}$. We define successor function $next(C, w_i)$ such that $next(C, w_i) = w_{i+1}$ for every $i \in [l-1]$ and $next(C, w_l) = w_1$. If $l = |\mathcal{D}|$, we say that counter is* space-optimal.

Definition 2 (Gray code) *A Gray code of length l over domain $\mathcal{D} = \mathcal{D}_1 \times \cdots \times \mathcal{D}_n$ is counter of length l over $\mathcal{D}$ such that w_i and $next(C, w_i)$ differ in exactly one coordinate. By allowing consecutive tuples to differ in at most c coordinates for some fixed constant $c \geq 1$, we get a c-*Gray code *or quasi-Gray code if c is not specified.*

Definition 3 (Decision Assignment Tree) *Let $\mathcal{D}^n$ be a domain and $\langle x_1, \dots, x_n \rangle$ be a variables for an elements of that domain.* Decision Assignment Tree *(DAT) is a rooted $|\mathcal{D}|$-ary tree such that each internal node is labeled by one of the x_i variables and each outgoing edges are labeled with a distinct element of $\mathcal{D}$ (these represent read operations on the input). Each leaf node is labeled by a set of assignment instructions assigning some fixed values to a subset of variables.*

The read complexity *of DAT T,* READ(T)*, is the maximum length of any path from root to the leaf. The* write complexity*,* WRITE(T)*, is the maximum number of assignments in any leaf node.*

Length	Read complexity	Write complexity
2^n	$n-1$	1
2^{n-1}	$\log n + 4$	4
$2^n - 2^{n-t}$	$\log n + t + 3$	2

An example of few known constructions of (quasi-)Gray codes over $\mathbb{Z}_2^n$.

Now, we can state the main results of the article (we will prove only the second one in this talk).

Theorem 4 *Let $m \in \mathbb{N}$ be odd and $n \in \mathbb{N}$ be sufficiently large. Then there is a space-optimal quasi-Gray code C over $\mathbb{Z}_m^n$ such that next(C, w) can be implemented by DAT with read complexity at most $4 \log_m n$ and write complexity 2.*

Theorem 5 *Let $n \in \mathbb{N}$ be sufficiently large. Then there is a quasi-Gray code C of length at least $2^n - 20n$ over $\mathbb{Z}_2^n$ such that next(C, w) can be implemented by DAT with read complexity at most $6 + \log n$ and write complexity 2.*

Chinese remainder theorem for counters

The first tool is based on the well-known Chinese remainder theorem. Using this tool, we can compose efficient counters for small domains into the counter for larger domain, while keeping the complexities reasonably small.

Theorem 6 (Chinese Remainder Theorem) *Let $n_1, \dots, n_k$ are pairwise co-prime integers, $N = \prod_{i=1}^{k} n_i$ and $a_1, \dots, a_k$ are arbitrary integers. Then there exists exactly one integer $0 \le x < N$ such that $x \equiv a_i \pmod{n}_i$ for every $i = 1, \dots, k$. In other words, there is a bijection (isomorphism in fact) between $\mathbb{Z}_N$ and $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2} \times \cdots \times \mathbb{Z}_{n_k}$.*

Theorem 7 (Chinese Remainder Theorem for Counters) *Let $r \in \mathbb{N}$ and C_i for $i \in [r]$ are counters of length l_i over $\mathcal{D}_i$ computed by DAT T_i. Moreover, let $l \ge r - 1$ and $l_2, \dots, l_r$ are pairwise co-prime. Then there exists a DAT T implementing counter C of length $\prod_{i=1}^{r} l_i$ over $\mathcal{D}_1 \times \cdots \times \mathcal{D}_r$. Furthermore $\mathrm{READ}(T) = n_1 + \max_{i=2}^{r}(\mathrm{READ}(T_i))$ and $\mathrm{WRITE}(T) = \mathrm{WRITE}(T_1) + \max_{i=2}^{r}(\mathrm{WRITE}(T_i))$, where n_1 number of variables in T_1.*

Permutations and decomposition of counters

This tool is based on the fact, that counters are basically nothing more than a cycles in permutations. By smart decomposition of the underlying permutation, we can design DAT with low complexity.

Lemma 8 *Let $\mathcal{D}$ be a domain, $\sigma_1, \dots, \sigma_k$ permutations on $\mathcal{D}$ such that $\sigma = \sigma_k \circ \sigma_{k-1} \circ \dots \sigma_1$ is cycle of length l and $T_1, \dots, T_k$ DATs implementing $\sigma_1, \dots, \sigma_k$. Let $\mathcal{D}'$ be domain and T' a DAT implementing a counter C' of length $k' \ge k$ over $\mathcal{D}'$.*

Then there exists a DAT T implementing a counter C of length $k'l$ over $\mathcal{D} \times \mathcal{D}'$ such that $\mathrm{READ}(T) = r' + \max_{i=1}^{k}(\mathrm{READ}(T_i))$ and $\mathrm{WRITE}(T) = \mathrm{WRITE}(T') + \max_{i=1}^{k}(\mathrm{WRITE}(T_i))$, where r' is number of variables in T'

Counters via linear transformation

Finally, we show the construction of a counter that gives the Theorem 5. The counter is defined through a repeated application of a particular linear mapping – represented by a matrix of full rank. By decomposing matrix into elementary matrices and using the lemma from previous section, we get the desired counter.

Lemma 9 *Every invertible matrix $A \in \mathbb{F}^{n \times n}$ can be written as product of at most $n^2 + 4(n-1)$ elementary matrices.*

Lemma 10 *Let $z \in \mathbb{Z}_q^n$ be a root of primitive polynomial and A be a matrix representing a mapping $x \mapsto z \cdot x$. Then $\mathbb{Z}_q^n = \{0, 1, z, z^2, \dots, z^{q^n-2}\}$ and A can be decomposed into at most $n + 4(n-1)$ elementary matrices.*

Jana Novotná
janca@kam.mff.cuni.cz
Presented paper by Peter A. Golovach, Pinar Heggernes, Dieter Kratsch

Enumeration and Maximum Number of Minimal Connected Vertex Covers in Graphs

(https://link.springer.com/chapter/10.1007/978-3-319-29516-9_20)

Introduction

Connected Vertex Cover is one of the classical problems of computer science. In the talk we first prove that the number of minimal connected vertex covers of a general graph is at most 1.8668^n and show an algorithm that enumerates these sets in time $O(1.8668^n)$. Second, we show that for chordal graphs there exist a tight bound on the maximum number of minimal connected vertex covers. And we show the relation between maximum numbers of minimal vertex cover and minimal connected vertex cover.

Definitions

We use the following notation. For a vertex v, we denote by $N_G(v)$ the (open) neighborhood of v, i.e., the set of vertices that are adjacent to v in G. The closed neighborhood is $N_G[v] = N_G(v) \cup \{v\}$. We use $G[U]$ to denote the subgraph of G induced by $U \subset V(G)$, we use $G - U$ to denote $G[V(G) \setminus U]$.

A set of vertices $U \subseteq V(G)$ is a *vertex cover* of G if for every $uv \in E(G)$, $u \in U$ or $v \in U$. A vertex cover U is *connected* if U is a connected set. A (connected) vertex cover U is *minimal* if no proper subset of U is a (connected) vertex cover. A set of vertices is an *independent set* if there is no edge between any pair of these vertices.

A vertex v is a *cut vertex* of a connected graph G if $G - v$ is disconnected.

A graph G is *chordal* if the length of a longest induced cycle in G is at most three.

Note 1 *U is a (minimal) vertex cover of G if and only if $V(G) \setminus U$ is a (maximal) independent set of G.*

Note 2 *If v is a cut vertex, then v belongs to every connected vertex cover.*

Theorems and algorithms

Theorem 3 *The maximum number of minimal vertex covers of an arbitrary graph is at most $3^{n/3}$, and these can be enumerated in time $O(3^{n/3})$.*

```
Algorithm: EnumMIS($S, F$)
Input: Two disjoint sets $S, F \subseteq V(G)$.
Output: Maximal independent sets of $G$ such that they contains $S$.
1. if $F = \emptyset$ then output $S$ if $S$ is a maximal independent set, stop;
2. if $\exists v$ in $F$ such that it has no neighbours in $F$ then EnumMIS($S \cup \{v\}, F \setminus \{v\}$);
3. else if $\exists v$ in $F$ such that it has exactly one neighbour $u$ in $F$ then
    EnumMIS($S \cup \{v\}, F \setminus \{u, v\}$), EnumMIS($S \cup \{u\}, F \setminus \{N_G[u]\}$)
4. else (every vertex $v$ in $F$ has at least two neighbours in $F$) pick any $v \in F$,
        for every vertex $u$ in $N_G[v]$ do EnumMIS($S \cup \{u\}, F \setminus N_G[u]$);
```

;

Theorem 4 *The maximum number of minimal connected vertex covers of an arbitrary graph is at most* 1.8668^n*, and these can be enumerated in time* $O(1.8668^n)$*.*

```
Algorithm: EnumCVC($S, F$)
Input: Two disjoint sets $S, F \subseteq V(G)$.
Output: Minimal connected vertex covers $U$ such that $S \subseteq U \subseteq S \cup F$.
1. if $S$ is a minimal connected vertex cover then return S and stop;
2. if $F = \emptyset$ then stop;
3. if there are two adjacent vertices $u, v \in F$ then branch as follows;

    1. select $u$, i.e., set $S' = S \cup \{u\}$, $F' = F \setminus \{u\}$, call EnumCVC($S', F'$)

    2. discard $u$ and select its neighbours, i.e., set $S' = S \cup N_G(u)$, $F' = F \setminus N_G[u]$, call
       EnumCVC($S', F'$)

4. if $F$ is an independent set then
    let $s$ be the number of components of $G[S]$. Consider every nonempty set $X \subseteq F$ of size
    at most $s - 1$ and output $S \cup X$ if $S \cup X$ is minimal connected vertex cover.
```

Theorem 5 *The maximum number of minimal connected vertex covers of a chordal graph is at most* $3^{n/3}$*, and these can be enumerated in time* $O(3^{n/3})$*.*

Michal Opler
opler@iuuk.mff.cuni.cz
Presented paper by Kasper Green Larsen and Jelani Nelson

Optimality of the Johnson-Lindenstrauss Lemma

(https://arxiv.org/abs/1609.02094)

Introduction

In modern algorithm design, often data is high-dimensional, and one seeks to first pre-process the data via some dimensionality reduction scheme that preserves geometry in such a way that is acceptable for particular applications. The lower-dimensional embedded data has the benefit of requiring less storage, less communication bandwith to be transmitted over a network, and less time to be analyzed by later algorithms.

A cornerstone dimensionality reduction result is the following Johnson-Lindenstrauss (JL) lemma [1].

Theorem 1 (JL lemma) *Let $X \subset \mathbb{R}^d$ be any set of size n, and let $\varepsilon \in (0, 1/2)$ be arbitrary. Then there exists a map $f : X \to \mathbb{R}^m$ for some $m = O(\varepsilon^{-2} \lg n)$ such that*

$$\forall x, y \in X,\ (1-\varepsilon)\left\|x-y\right\|_2^2 \leq \left\|f(x)-f(y)\right\|_2^2 \leq (1+\varepsilon)\left\|x-y\right\|_2^2 . \tag{2}$$

Main Result

In this paper, the authors settle the optimality of the JL lemma for almost the full range of ε.

Theorem 2 *For any integers $n, d \geq 2$ and $\varepsilon \in (\lg^{0.5001} n/\sqrt{\min\{n, d\}}, 1)$, there exists a set of points $X \subset \mathbb{R}^m$ of size n such that any map $f : X \to \mathrm{R}^m$ providing the guarantee (2) must have*

$$m = \Omega(\varepsilon^{-2} \lg(\varepsilon^2 n)).$$

Ingredients

Let $e_1, \ldots, e_d$ denote the standard unit vectors in R^d. For any set $S \subset [d]$ of $k = \varepsilon^{-2}/256$ indices, define a vector $y_S = \sum_{j \in S} e_j/\sqrt{k}$. Let $Q = n - d - 1$. For every Q-tuple γ of sets $S_1, \ldots S_Q \subset [d]$ define a point sequence P_γ as $(0, e_1, \ldots, e_d, y_{S_1}, \ldots, y_{S_Q})$. Our goal is to show that images of these sequences must be very different via an encoding argument.

Lemma 3 *Let T be an origin symmetric convex body in $\mathbb{R}^m$. For any $0 < \varepsilon < 1$, one can cover T using at most $2^{m \lg(1+2/\varepsilon)}$ translated copies of εT.*

Bibliography

[1] Johnson, William and Lindenstrauss, Joram. *Extensions of Lipschitz maps into a Hilbert space.* Contemporary Mathematics, 26:189–206, 1984.

Giacomo Paesani

giacomo.paesani@durham.ac.uk

Presented paper by M.Johnson, G.Paesani, D.Paulusma

Connected Vertex Cover is Polynomial-Time Solvable For $sP_1 + P_5$-Free Graphs

(https://arxiv.org/pdf/1712.08362v2.pdf)

Introduction

Definition 1 *A subset S of V is said to be a connected vertex cover if it is a vertex cover and S induces a connected subgraph.*

The Connected Vertex Cover problem is to determine a minimum vertex cover of a graph G that induces a connected subgraph.

Definition 2 *Let H be a graph, then another graph G is H-free if in G there is no induced subgraph isomorphic to H.*

Definition 3 *A linear forest is a graph formed from the disjoint union of paths.*

This is a well-studied problem, known to be NP-complete for restricted graph classes, and, in particular, for H-free graphs if H is not a linear forest. We prove that Connected Vertex Cover is polynomial-time solvable for $sP_1 + P_5$-free graphs for each integer $s \geq 0$, providing an explicit algorithm.

Bibliography

[1] M. Johnson, G. Paesani and D. Paulusma, *Connected Vertex Cover for $sP_1 + P_5$-Free Graphs*, https://arxiv.org/pdf/1712.08362.pdf, 2018.

Sophie Rehberg
sophierehberg@mailbox.tu-berlin.de

Presented paper by Ileana Streinu (2005)

Pseudo-Triangulations, Rigidity and Motion Planning

(http://linkage.cs.umass.edu/publications.html)

Introduction

The paper "proposes a combinatorial approach to planning non-colliding trajectories for a polygonal bar-and-joint framework. It is based on a new class of simple motions induced by expansive one-degree-of-freedom mechanisms, which guarantee noncollisions by moving all points away from each other. Their combinatorial structure is captured by pointed pseudo-triangulations, a class of embedded planar graphs for which we give several equivalent characterizations and exhibit rich rigidity theoretic properties. The main application is an efficient algorithm for the Carpenter's Rule Problem: convexify a simple bar-and-joint planar polygonal linkage using only non-self-intersecting planar motions." (Streinu, 2005)

Pseudo Triangulations

Definition 1 *A vertex in an embedded graph is called **pointed**, if all edge vectors around that vertex are strictly contained in one half-plane. A **pointed graph** is an embedding (in R^2), such that all vertices are pointed. A **simple polygon** is a planar embedding of a cycle graph.*

Definition 2 *A **pseudo-triangle** is a simple polygon with exactly 3 convex vertices (corners).*

Definition 3 *A **pseudo-triangulation** is a graph-embedding whose outer face is convex and all interior faces are pseudo-triangles.*

*A **minimum-pseudo-triangulation** has the least number of edges among all pseudo-triangulations on the same point set.*

*A **pointed pseudo-triangulation** is pointed as a graph.*

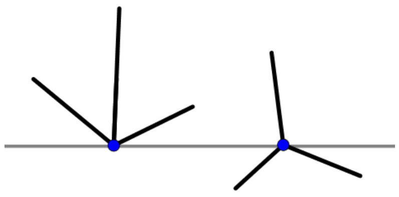


Figure 8: Pointed and non-pointed vertex.

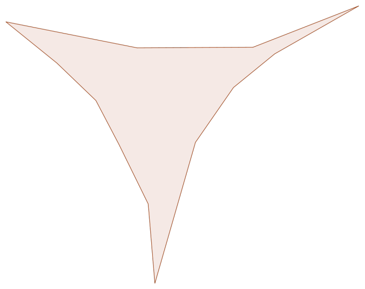


Figure 9: Pseudo triangle (three convex corners)

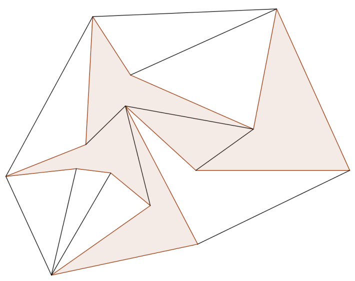


Figure 10: Pseudo Triangulation, that is pointed.

Theorem 4 (Characterisation of Pseudo-Triangulations) *$G = (V, E)$ an embedded graph, then the following are equivalent:*

1. *G is a minimum-pseudo-triangulation.*
2. *G is a pointed pseudo-triangulation.*

3. *G is a pseudo-triangulation with $2n-3$ edges (and equivalently $n-2$ faces).*

4. *G is planar, pointed and has $2n-3$ edges.*

5. *G is planar, pointed and maximal (w.r.t. edge-inclusion) with that property.*

Moreover, if one of the above conditions holds, then G is a Laman-Graph.

Definition 5 *A* ***Laman-minus-one graph****, is a Laman-graph, with one edge removed.*

Subgraphs, that induce exactly $2n'-3$ edges, are called ***r-components****. (These are Laman-graphs themselves.)*

Lemma 6 *The edge set of a Laman-minus-one graph is partitioned into (disjoint) r-components.*

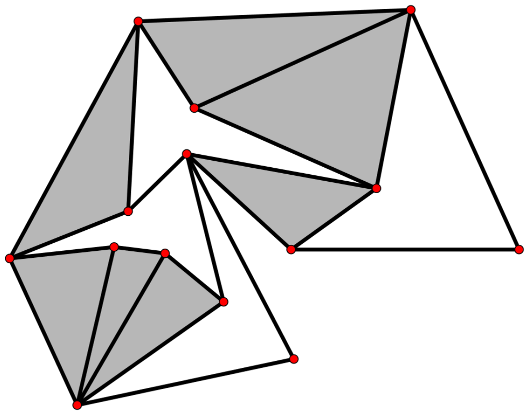


Figure 11: A Laman-minus-one graph and its r-components

Rigidity

Definition 7 *A* ***(bar-and-joint) framework*** *(G, L) is a graph $G = (V, E)$ together with a set of strictly positive weights $L = \{\ell_e : e \in E\}$ (edge lengths).*

An ***realisation*** *$G(\mathcal{P})$ of (G, L) on a set of points $\mathcal{P} = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ is a mapping $i \mapsto p_i$ of vertices to points, s.t. $\|p_i - p_j\| = \ell_e$, where $e = (ij)$.*

Definition 8 *The set of all possible realisations of a framework $p = (x_1, y_1, \ldots, x_n, y_n) \in \mathbb{R}^{2n}$ (up to rigid motion), is called its* ***configuration space***

The dimension of the component of the configuration space, containing a framework (realisation) $G(\mathcal{P})$ is called ***number of degrees of freedom****.*

If that is component an isolated point, $G(P)$ is called ***rigid****, otherwise it is* ***flexible****.*

Definition 9 *A rigid framework is called* ***minimal****, if removing any edge would make it flexible.*

Definition 10 *$v = (v_1, \ldots, v_n)$ is called* ***infinitesimal movement****, if*

$$\langle p_i - p_j \ , \ v_i - v_j \rangle = 0$$

A framework is called ***infinitesimal flexible****, if there is a non-trivial infinitesimal movement, otherwise it is called* ***infinitesimal rigid****.*

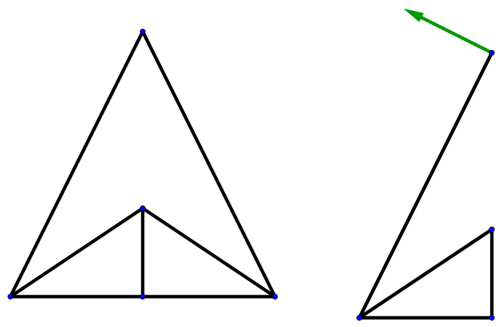


Figure 12: Framework with a rigid and a flexible embedding (due to symmetry).

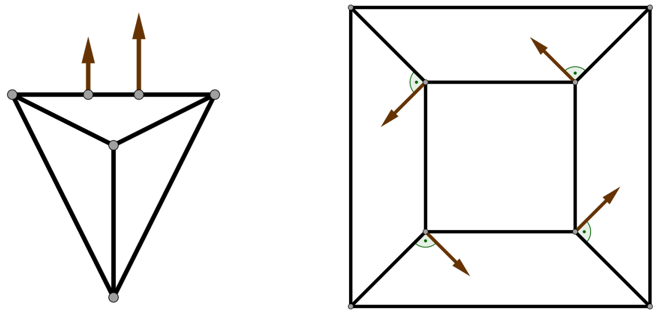


Figure 13: Rigid frameworks which have an infinitesimal motion

Edge Map and Rigidity Matrix

Edge map $f_G : \mathbb{R}^{2n} \mapsto \mathbb{R}^m$

$$f_G(p_1, \dots, p_n) = (\|p_i - p_j\|^2)_{ij \in E}$$
$$f_G(p_{1_x}, p_{1_y}, \dots, p_{n_x}, p_{n_y}) = ((p_{i_x} - p_{j_x})^2 + (p_{i_y} - p_{j_y})^2)_{ij \in E}$$

Rigidity matrix

$$M := \frac{1}{2} df_G(p) = \begin{array}{c} \\ ij \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & \dots & i & \dots & j & \dots & n \end{array} \\ \begin{pmatrix} & \dots & & \dots & & \dots & \\ 0 & \dots & p_i - p_j & \dots & p_j - p_i & \dots & 0 \\ & \dots & & \dots & & \dots & \end{pmatrix} \end{array}$$

$$= \begin{array}{c} \\ ij \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccccccc} 1_x & 1_y & \dots & i_x & i_y & \dots & j_x & j_y & \dots & n_x & n_y \end{array} \\ \begin{pmatrix} & & \dots & & & \dots & & & \dots & & \\ 0 & 0 & \dots & p_{i_x} - p_{j_x} & p_{i_y} - p_{j_y} & \dots & p_{j_x} - p_{i_x} & p_{j_y} - p_{i_y} & \dots & 0 & 0 \\ & & \dots & & & \dots & & & \dots & & \end{pmatrix} \end{array}$$

Subspace of infinitesimal movement

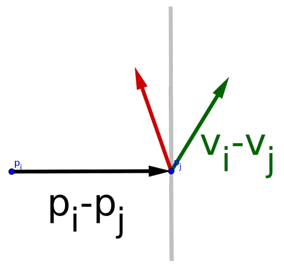
$$\ker M = \{v \in \mathbb{R}^{2n} : Mv = (\langle p_i - p_j, v_i - v_j \rangle)_{ij \in E} = 0\} \subset \mathbb{R}^{2n}$$

Modification factor

$$d_{ij} = \langle p_i - p_j, v_i - v_j \rangle \text{ for } ij \in E$$

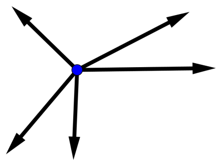
then

$$\operatorname{Im} M = \{d \in \mathbb{R}^m : d = Mv\} \subset \mathbb{R}^m$$



A **Self-Stress** on a frameworks $G(\mathcal{P})$ is an assignment of scalars $w_{ij} \in \mathbb{R}$ to edges $ij \in E$, s.t.

$$\forall i \in V : \sum_{ij \in E} w_{ij}(p_i - p_j) = 0.$$



Then the subspace of self-stress vectors is

$$\ker M^T = \{w \in \mathbb{R}^m : M^T w = 0\} \subset \mathbb{R}^m$$

By the fundamental theorem of linear algebra we know:

$$\begin{aligned}\operatorname{Im} M &\perp \ker M^T \\ \ker M &\perp \operatorname{Im} M^T \\ \dim\ker M + \dim\operatorname{Im} M^T &= 2n \\ \dim\operatorname{Im} M + \dim\ker M^T &= m \\ \dim\operatorname{Im} M &= \dim\operatorname{Im} M^T\end{aligned}$$

Observe: Vector space of infinitesimal movements always contains at least translations and rotations, so $\dim\ker M \geq 3$.

Lemma 11 *Let $m = 2n-3$. Then* $\dim\ker M = 3 \;\Leftrightarrow\; \dim\ker M^T = 0$.

In that case the framework $G(\mathcal{P})$ is infinitesimal rigid, iff it has no self-stress.

Lemma 12 *If $m \leq 2n-4$, then there exists a non-trivial infinitesimal movement.*

Lemma 13 *If $m \geq 2n-2$, then there exists a non-trivial self-stress.*

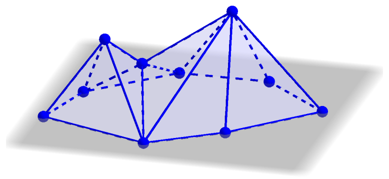
Theorem 14 (1dof-Mechanism) *Let G be a Laman-minus-one graph (i.e. $m = 2n-4$) with a generic (i.e. regular and self-stress-free) realisation $G(\mathcal{P})$.*

Then the corresponding component of its configuration space is one-dimensional and smooth in the neighbourhood of the given generic embedding.

Definition 15 *A framework satisfying the conditions of the theorem is called a **1dof-mechanism** (one degree of freedom mechanism).*

Self-Stress and Maxwell-Lifting

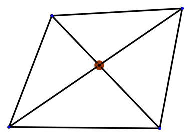
Definition 16 *A **3-dim lifting** of a planar framework $G(\mathcal{P})$ is an assignment of a "height" $z_i \in \mathbb{R}$ to each vertex $i \in V$, s.t. after "lifting" the graph the faces are planar.*



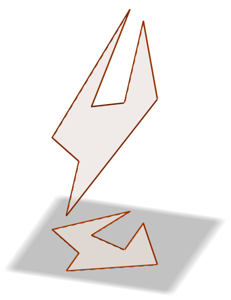
Theorem 17 (Maxwell 1864) *A planar framework has a non-trivial 3-dimensional lifting, if and only if it has non-trivial self-stress.*

Moreover, the lifting maps edges with positive self-stress to mountain-edges, edges with negative self-stress to valley-edges and edges with zero self-stress to flat edges.

Bows construction: For every crossing introduce a new vertex and divide edges. That construction preserves all self-stress properties.



In every non-trivial 3-dim lifting of an planar polygon all vertices of maximal height are convex vertices (corners).



Theorem 18 *Pointed pseudo-triangulations are always infinitesimal rigid.*

Expansion Signature

Definition 19 *A 1dof mechanism is called* ***infinitesimal expansive****, if there is an infinitesimal movement s.t. all free diagonals are expanded.*

Recall:

an infinitesimal movement v expands a diagonal ij $\quad\Leftrightarrow\quad$ $d_{ij} = \langle p_i - p_j, v_i - v_j \rangle > 0$

$s = (s_{ij})_{ij \in E}$ with $s_{ij} := \operatorname{sign} d_{ij}$ is called **expansion signature**.

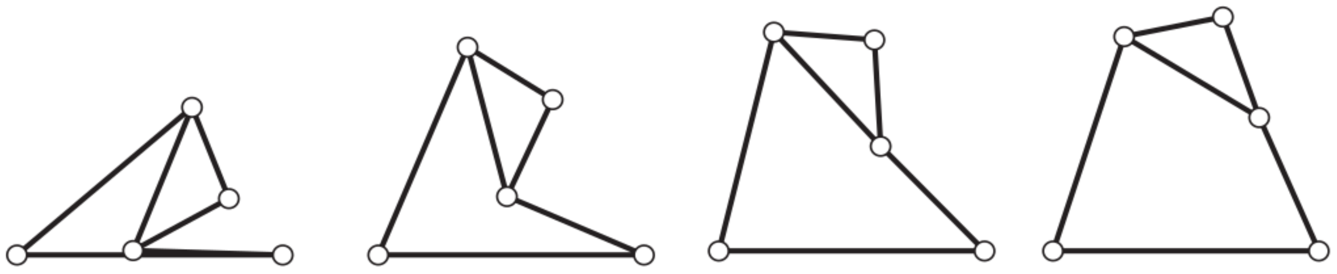


Figure 14: Expansion signature s_{ij} is constant on an open interval, changes only in singularities and whenever d_{ij} becomes zero.

Want to show: Laman-minus-one graph 1dof mechanism is expansive.

Therefor consider relation between modification factor d_{ij} and self-stress w_{ij}.

Consider:

$$\text{self-stress subspace} \quad \ker M^T \ \perp \ \operatorname{Im} M \quad \text{image space}$$

It follows $\forall w = (w_{ij}) \in \mathbb{R}^m$ and $\forall v \in \mathbb{R}^{2n}$

$$0 = \langle w, Mv \rangle = \sum_{ij \in E} w_{ij} \langle p_i - p_j, v_i - v_j \rangle = \sum_{ij \in E} w_{ij} d_{ij}$$

So, in the non-trivial case there exist edges (ij) and (kl), s.t. $w_{ij} d_{ij} < 0 \ \Leftrightarrow \ w_{kl} d_{kl} > 0$. From that the next Lemma follows (one can also derive it from Farkas' Lemma or programming duality).

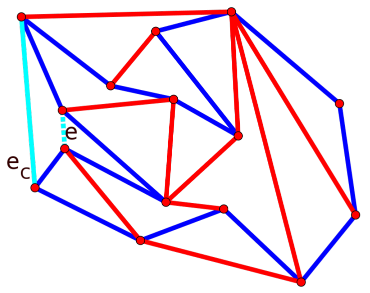
Lemma 20 (Farkas' Lemma for self-stress) *Let $G(\mathcal{P})$ be a realisation of a Laman graph with an additional diagonal e and let v be an infinitesimal movement, which maintains edge length of $2n-4$ edges and infinitesimally changes the ones of e, e_c.*

Then $G(\mathcal{P})$ has a self-stress and the signatures of the two flexible edges (e, e_c) fulfil one of the following conditions:

1. *$w_e \cdot w_{e_c} > 0$ and $d_e \cdot d_{e_c} < 0$*

   or

2. *$w_e \cdot w_{e_c} < 0$ and $d_e \cdot d_{e_c} > 0$*



Theorem 21 *The framework of a pseudo-triangulation, where one edge of the convex hull was removed, is an infinitesimal 1dof* expansive *mechanism.*

Algorithm

Input: planar polygon on set of points $\mathcal{P}$

Output: convexified polygon; sequence of movements, given by pseudo-triangulations with one edge removed

1. compute initial pseudo-triangulierung on $\mathcal{P}$ and remove arbitrary edge on convex hull, which is not a polygon edge
2. Repeat until polygon is convex:
   (a) Pin down arbitrary edge. Move 1dof mechanism in expansive direction, until an alignment event occurs
   (b) perform a "Flip" or "Freeze" operation, reconfigure (if necessary) the pseudo-triangulation and continue.

There are 3 **types of alignment events**:

Freeze: Two adjacent edges of the polygon become collinear. Then freeze the joint, eliminating one vertex of the polygon.

Flip: Two adjacent added diagonals or one diagonal and an edge of the polygon become collinear. Then perform a flip in the pseudo-triangulation to obtain a pseudo-triangulation with a semi-simple face.

False: The missing convex hull edge aligns with another edge. Then continue the motion and do not consider this a proper event.

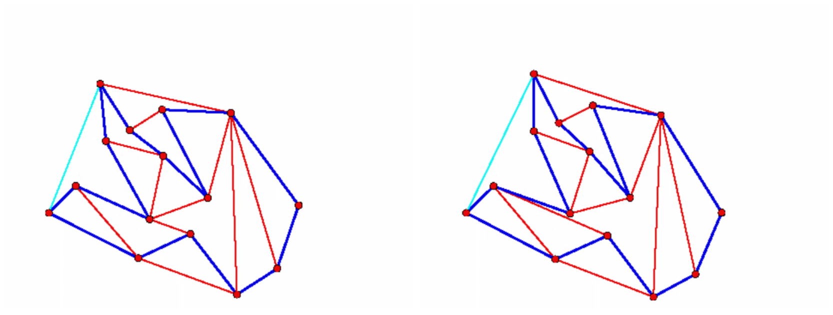


Figure 15: A flip event

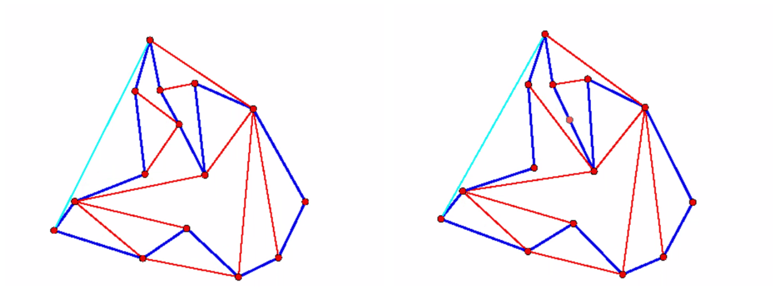


Figure 16: A freeze event

Jakub Svoboda

jakubs@kam.ms.mff.cuni.cz

Presented paper by Timothy M. Chan

Approximation Schemes for 0-1 Knapsack

(https://pdfs.semanticscholar.org/7c66/6b3d8197b52b836ca876d33d67731c4efd72.pdf)

Introduction and definitions

We revisit the standard 0-1 knapsack problem. This is well known NP hard problem, but we can create fully polynomial-time approximation scheme. In the talk, you will see simple algorithm with approximation factor $1+\varepsilon$ and running time near $\mathcal{O}(n + \left(\frac{1}{\varepsilon}\right)^{\frac{5}{2}})$.

Definition 1 *In the 0-1 knapsack problem, we are given a set of n items where the i-th item has weight $w_i \leq W$ and profit $p_i > 0$, and we want to select a subset of items with total weight bounded by W while maximizing the total profit. In other words, we want to maximize $\sum_{i=1}^{n} p_i\zeta_i$ subject to the constraint that $\sum_{i=1}^{n} w_i\zeta_i \leq W$ over all $\zeta_1, \zeta_2, \dots, \zeta_n \in \{0,1\}$.*

Definition 2 *Fully polynomial-time approximation schemes (FPTASs) are algorithms with approximation factor $1+\varepsilon$ for any given parameter $\varepsilon \in (0,1)$, taking time polynomial in n and $\frac{1}{\varepsilon}$.*

Building blocks

Observation 3 *We may discard all items with $p_i \leq \frac{\varepsilon}{n}\max_j p_j$.*

We can round all items to power of $1+\varepsilon$.

Definition 4 *Let function*

$$f_i(x) = \max\left\{\sum_{i=1}^{n} p_i\zeta_i : \sum_{i=1}^{n} w_i\zeta_i \leq x, \zeta_1, \zeta_2, \dots, \zeta_n \in \{0,1\}\right\}$$

for all $x \in \mathbb{R}$ be knapsacking function.

Definition 5 *We say that a step function is p-uniform if the function values are of the form $-\infty, 0, p, 2p, \dots, pl$ for some l.*

Furthermore, we say that a p-uniform function is pseudo-concave if the sequence of differences of consecutive x-breakpoints is nondecreasing.

Lemma 6 *Let f and g be monotone step functions with total complexity $\mathcal{O}(l)$ (i.e., with $\mathcal{O}(l)$ steps). We can compute $f \oplus g$ in*

- *$l^2/2^{\Omega(\sqrt{\log l})}$ time if f and g are p-uniform.*
- *$\mathcal{O}(l)$ time if f is p-uniform and g is p-uniform and pseudo-concave.*
- *$\mathcal{O}((l + l' \cdot \frac{p'}{p})\log\frac{p'}{p})$ time if f is p-uniform, and g is p'-uniform and pseudo concave with complexity l' and p' is multiple of p.*

Lemma 7 *Let f and g be monotone step functions with total complexity and ranges contained in $\{-\infty, 0\} \cup [A,B]$. Then we can compute a monotone step function that approximates $f \oplus g$ with factor $1+\mathcal{O}(\varepsilon)$ and complexity $\mathcal{O}(\frac{1}{\varepsilon})$ in*

- *$\mathcal{O}(l + \left(\frac{1}{\varepsilon}\right)^2/2^{\Omega\sqrt{\log\frac{1}{\varepsilon}}})$ time in general.*

- *$\mathcal{O}(l + \frac{1}{\varepsilon})$ time if g is p-uniform and pseudo-concave.*

Lemma 8 *Let $f_1, f_2, \ldots, f_m$ be monotone step functions with total complexity $\mathcal{O}(n)$ and ranges contained in $\{-\infty, 0\} \cup [A, B]$. Then we can compute a monotone step function that approximates $f_1 \oplus \cdots \oplus f_m$ with complexity $\mathcal{O}(\frac{1}{\varepsilon})$ in*

- *$\mathcal{O}(l + \left(\frac{1}{\varepsilon}\right)^2 m/2^{\Omega(\sqrt{\log \frac{1}{\varepsilon}})})$ time in general.*
- *$\mathcal{O}(l + \frac{1}{\varepsilon} m^2)$ time if g is p-uniform and pseudo-concave.*

Final touch

Theorem 9 *Let $f_1, \ldots, f_m$ be monotone step functions with total complexity $\mathcal{O}(n)$ and ranges contained in $\{-\infty, 0\} \cup [A, B]$. If every f_i is p_i-uniform and pseudo-concave for some p_i, then we can compute a monotone step function that approximates $f_1 \oplus \cdots \oplus f_m$ with factor $1 + \mathcal{O}(\varepsilon)$ and complexity $\mathcal{O}(\frac{1}{\varepsilon})$ in $\mathcal{O}\left(n + \left(\frac{1}{\varepsilon}\right)^{\frac{3}{2}} m/2^{\Omega(\sqrt{\log \frac{1}{\varepsilon}})}\right)$ time.*

Jakub Tětek
j.tetek@gmail.com
Presented paper by Tsvi Kopelowitz, Ely Porat

A Simple Algorithm for Approximating the Text-To-Pattern Hamming Distance

(http://drops.dagstuhl.de/opus/volltexte/2018/8308/)

Definitions

Let $h : \Sigma \to \Sigma'$ and S a string such that $S = s_1 s_2 \cdots s_n$. Then $h(S) = h(s_1)h(s_2)\cdots h(s_n)$.
T – the text.
P – the pattern.
n – the length of T.
m – the length of P.
$T_i = T[i, \cdots, i+m-1]$

The Hamming distance $H(s,t)$ of strings s, t is the number of aligned character mismatches netween the two strings.

Results

Goal: find i, such that the Hamming distance between T_i and P is smallest possible.

Proven in the paper:

Theorem 1 *There exists an algorithm that with high probability solves the pattern-to-text approximate Hamming distance problem and runs in $O(\frac{n}{\varepsilon}\log n \log m)$ time.*

Importent theorem used in the paper:

Theorem 2 *[1] Given a binary text T of size n and a binary pattern P of size m, there exists an $O(n \log m)$ time algorithm that computes for all locations i in T the number of times that a 1 in T_i is aligned with a 1 in P.*

```
ApproxHAM($T_j$, $P$, $\varepsilon$)
for $i = 1$ to $c \log n$

do Pick a random $h : \Sigma \to \{1, 2, \cdots, \frac{2}{n}\}$.

compute $x_i = HAM(h(T_j), h(P))$
return $max_{1 \le i \le c \log n}\{x_i\}$
```

Bibliography

[1] M.J. Fischer and M.S. Paterson *String matching and other products.* Complexity of computation. In SIAM–AMS Proceedings, vol. 7, page 113–125, 1974.

Pablo Oviedo Timoneda

pablo_ovitim@hotmail.es

Presented paper by P. Schweitzer, P. Schweitzer

Minimal Asymmetric Graphs

(https://arxiv.org/pdf/1605.01320.pdf)

Abstract

In this talk we show that up to isomorphism there are exactly 18 minimal asymmetric graphs. We also show that these graphs are exactly the minimal involution-free graphs.

Introduction

In 1998 Nešetřil conjectured that there exists only a finite number of minimal asymmetric graphs. Since then Nešetřil and Sabidussi showed that there are exactly nine minimal asymmetric graphs containing P_5, the path of length 4, as an induced subgraph and identified 18 minimal asymmetric graphs in total. They also conjectured that the set of minimal asymmetric graphs and set of minimal involution-free graphs are the same.

In 2017, Pascal Schweitzer and Patrick Schweitzer confirmed the two conjectures showing that there are exactly 18 minimal involution-free graphs and that the minimal involution-free graphs are exactly the minimal asymmetric graphs.

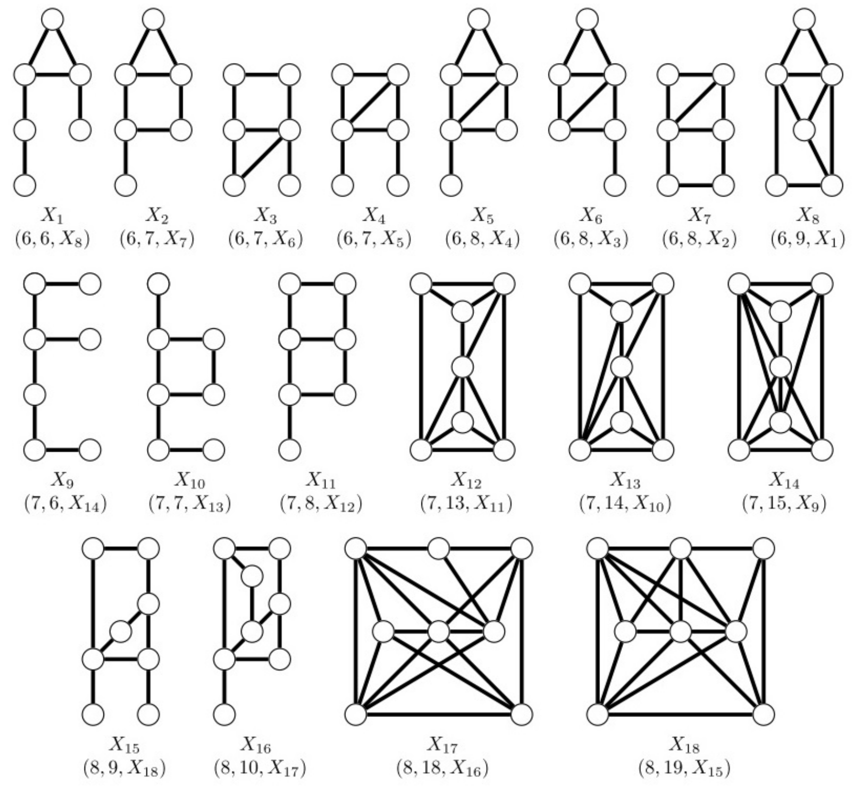

X_1 $(6, 6, X_8)$
X_2 $(6, 7, X_7)$
X_3 $(6, 7, X_6)$
X_4 $(6, 7, X_5)$
X_5 $(6, 8, X_4)$
X_6 $(6, 8, X_3)$
X_7 $(6, 8, X_2)$
X_8 $(6, 9, X_1)$
X_9 $(7, 6, X_{14})$
X_{10} $(7, 7, X_{13})$
X_{11} $(7, 8, X_{12})$
X_{12} $(7, 13, X_{11})$
X_{13} $(7, 14, X_{10})$
X_{14} $(7, 15, X_9)$
X_{15} $(8, 9, X_{18})$
X_{16} $(8, 10, X_{17})$
X_{17} $(8, 18, X_{16})$
X_{18} $(8, 19, X_{15})$

Fig. 1. The 18 minimal asymmetric graphs. These are also the minimal involution-free graphs. For each graph the triple $(n, m, \text{co-}G)$, describes the number of vertices, edges and the name of the complement graph, respectively. The graphs are ordered first by number of vertices and second by number of edges.

Martin Töpfer

mtopfer@gmail.com

Presented paper by Dana Angluin, James Aspnes, David Eisenstat

A Simple Population Protocol for Fast Robust Approximate Majority

(https://link.springer.com/article/10.1007/s00446-008-0059-z)

Abstract

Population protocol is a model for distributed computing in which randomly-chosen agents with little computational power interact with each other. This model can describe a network of very simple sensors, chemical reactions or can be implemented by DNA computations. This paper discuss the majority problem – whether there are more X's or Y's in the initial configuration.

Population Protocols

A *population protocol* consists of a set of n agents, each in some initial state. A *configuration* is the global state of the algorithm – it describes number of agents in each state. In each step of the computation two agents are picked uniformly at random and their states are changed according to the algorithm. The goal is to achieve convergence to some configuration, which represents the output, with high probability.

Complexity of an algorithm can be meassured in *parallel time* (number of iterations divided by n) and *space* (number of states of agents).

Byzantine agents models agents with some kind of error. More formally, Byzantine agent can simulate any state in an interaction. The chosen state can depend on the global configuration as well as on the state of the second chosen agent.

Algorithm for Majority

The algorithm uses just three states - X, Y and B, where B are blank/undecided agents. The rules are as follow:

$$\begin{aligned} X + Y &\to X + B \\ Y + X &\to Y + B \\ X + B &\to X + X \\ Y + B &\to Y + Y. \end{aligned}$$

Theorem 1 *Algorithm converges to all X's or all Y's in $\mathcal{O}(n \log n)$ w.h.p.*

Theorem 2 *If the difference between initial majority and minority populations is $\omega(\sqrt{n}\log n)$, the algorithm converges to correct output w.h.p.*

Theorem 3 *The algorithm can tolerate $\mathcal{O}(\sqrt{n})$ Byzantine agents and converge in $\mathcal{O}(n \log n)$ w.h.p. (there still could be a few agents in a wrong state – for example the Byzantine ones)*

Tomáš Toufar
toufi@iuuk.mff.cuni.cz
Presented paper by Benjamin Bergougnoux, Eduard Eiben, Robert Ganian, Sebastian Ordyniak, M.S. Ramanujan

Towards a Polynomial Kernel for Directed Feedback Vertex Set

(http://drops.dagstuhl.de/opus/volltexte/2017/8112/pdf/LIPIcs-MFCS-2017-36.pdf)

Introduction

In this paper authors study the Directed Feedback Vertex Set (DFVS for short) problem. The DFVS is one of the central problems in parameterized complexity; many questions regarding DFVS were major research problems that resisted solving for a long time or still stand open. One of remaining open questions is whether DFVS admits a polynomial kernel. The authors make a progress towards answering the question by showing an existence of a polynomial kernel for a weaker parameterization.

Parameterized complexity basics

In parameterized complexity we measure the running time not only with respect to the size of input but also with respect to some *parameter* (think of: size of the solution, treewidth of input graph...). Formally, *parameterized problem* is a language $\Pi \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a finite alphabet. A problem is *fixed-parameter tractable* if it admits an algorithm which decides whether an instance (I, k) is in Π in time $f(k)\text{poly}(|I|)$ for some computable function f.

A *kernelization* for a problem Π is a polynomial-time algorithm that given an instance (I, k) returns an instance (I', k') such that

- $(I, k) \in \Pi$ if and only if $(I', k') \in \Pi$ and
- $|I'| + k' \leq g(k)$, where g is a computable function.

The function g is referred to as the *size* of the kernel.

Notation

To distinguish between directed and undirected edges, directed edges will be referred to as *arcs*. If D is a digraph, we denote by $\overline{D}$ the underlying undirected graph. Note that if D has arcs (u, v) and (v, u) then $\overline{D}$ will contain two parallel u-v edges. The size of the undirected feedback vertex set S on input is denoted by k.

Directed Feedback Vertex Set parameterized by FVS (DFVS[FVS])
Input: A directed graph D, an integer p and a set S such that S is an undirected feedback vertex set of $\overline{D}$.
Parameter: $|S|$
Task: Determine whether there is a directed feedback vertex set of D of size at most p

Theorem 1 *The here is a kernel of $O(k^4)$ vertices for* DFVS[FVS].

Reduction Rule 1 *If a vertex v has a loop, delete v and decrease p by one.*

Reduction Rule 2 *If a vertex v is a source or a sink, delete v.*

Reduction Rule 3 *If an arc e is incident to a vertex with outdegree 1 or indegree 1, contract e.*

Reduction Rule 4 *Let u and v be two (not necessarily distinct) vertices in S such that there are at least $k+1$ internally vertex-disjoint directed u-v paths in D. Then add an arc from u to v.*

Definition 2 *Let (u,v) be an ordered pair of vertices in S if $u \neq v$, we refer to (u,v) as a* potential arc *in $D[S]$ and if additionally $(u,v) \notin D$ then we refer to (u,v) as* non-arc. *If on the other hand $u = v$, we refer to (u,v) as* self-loop.

We say that a vertex v contributes *to a potential arc or self-loop (u,w), if $(u,v) \in E(D)$ and $(v,w) \in E(D)$.*

We now partition the vertices of $D \setminus S$ in terms of their total degree in $D \setminus S$. Let A_0, A_1, A_2, and $A_{\geq 3}$ be the sets of all vertices in $D \setminus S$ that have total degree $0, 1, 2$, and at least 3, respectively, in $D \setminus S$. Furthermore, we denote by A'_i the set of vertices in A_i that do not contribute to some self-loop of $D[S]$.

Reduction Rule 5 *If v does not contribute to a non-arc of $D[S]$, then we remove v from D.*

Reduction Rule 6 *If $v \in A'_1$ does not contribute to a non-arc of $D[S]$, then*

- *if v is a source in $D \setminus S$, then we delete all arcs from v to S,*
- *if v is a sink in $D \setminus S$, then we delete all arcs from S to v.*

Definition 3 *Let $P = (v_1, \ldots, v_r)$ be a directed path of maximum length in $D \setminus S$ whose internal vertices are in A_2 Then we say that P is a* path segment *in $D \setminus S$. We further say that P is an* outer path segment *if at least one of its endpoints is not in A_2; otherwise we say that P is an* inner path segment.

We say that a path segment P contributes *to a potential arc (s, s') of $D[S]$ if there are i, j with $1 \leq i \leq j \leq r$ such that $(s, v_i) \in E(D)$ and $(v_j, s') \in E(D)$.*

Reduction Rule 7 *If an inner path segment does not contribute to a non-arc or a self-loop of $D[S]$, we remove all internal vertices of P.*

Definition 4 *Let $s \in S$ and let $P = (v_1, \ldots, v_r)$ be an induced directed path in $D \setminus S$, whose internal vertices are in A_2 and that satisfies:*

- *$(s, v_1) \in E(D)$ and $(s, v_r) \in E(D)$ and v_1 is a balanced vertex in A_2,*
- *for every i with $1 < i < r$ it holds that $(s, v_i) \notin E(D)$.*

Such a P is called an out-segment *for s. We say that P* contributes *to a potential arc or a self-loop (s, s') in $D[S]$ if there is an index i with $1 \leq i < r$ such that $(v_i, s') \in E(D)$.*

Reduction Rule 8 *Let $s \in S$ and let $P = (v_1, \ldots, v_r)$ be an out segment for s. If P does not contribute to a non-arc or a self-loop of $D[S]$, we remove the arc (s, v_1).*

Lukáš Vik
lukas.vik@icloud.com
Presented paper by Florian Lehner
The Firefighting on Trees and Cayley Graphs
(https://arxiv.org/pdf/1707.01224.pdf)

Introduction

Autor presents results for Hartnell's firefighter problem on infinite trees and Cayley graphs with connection to branching number of tees. We also draw some connections with paper by Danny Dyer, Eduardo Martinez.Pedroza and Brandon Thorne on geometry of firefighter problem referenced in the main paper.

Preliminaries: Oriented graphs, Cayley graphs, locally finite graphs, flows in graphs.

We now state some basic definitions from the paper for better understanding of given problem, because in the lecture we want to draw same other connection between the papers.

Definitions

Firefighter problem: Let $G = (V, E)$ be locally finite graph and f_n be sequence of positive integers. We are given finite set X_0 of vertices. These vertices are labeled as burning. At every step n, we can label set $|V_n| \leq f_n$ of verticies which are not burning as guarded. After this all adjacent vertices to burning vertices are labeled burning. This process stops when the set of all burning vertices does not change.

f_n-containment If there exists finite number of steps after which process stops for every finite set X_0, then we say that graph G satisfies *f_n-containment.* We say that graph G satisfies *exponential containment of rate λ* if there is $f_n = O(\lambda^n)$.

ball of radius k We define *ball of radius k with center r* as a set $B_r(k) := \{v \in V; |v| \leq k\}$, where $|v|$ is distance from r.

branching number For a tree rooted at r define

$$brT := sup\left\{\lambda; \exists \texttt{ non-zero flow } \theta \texttt{ from } r \texttt{ to } \infty : \theta(e) = \lambda^{-|e|}\right\},$$

where $|e|$ is distance of edge e from r.

Bibliography

[1] Florian Lehner. *Firefighting on trees and Cayley graphs* arxiv.org/abs/1707.01224

[2] D. Dyer, E. Martínez-Pedroza, and B. Thorne. *The coarse geometry of Hartnell's firefighter problem on infinite graphs.* Discrete Math, 2007.

List of participants

Kateřina Altmanová

Mikhail Beliayeu

Václav Blažej

Pavel Dvořák

Jiří Fiala

Tomáš Gavenčiak

Petr Glivický

Petr Gregor

Milan Hladík

Pavel Hubáček

Miloš Chromý

Lars Jaffke

Tereza Klimošová

Anna Kompišová

Karel Král

Jan Matyáš Křišťan

Robert Lukoťka

Tomáš Masařík

Ondřej Mička

Jana Novotná

Michal Opler

Giacomo Paesani

Daniël Paulusma

Sophie Rehberg

Robert Šámal

Veronika Slívová

Jakub Svoboda

Martin Tancer

Jakub Tětek

Pablo Oviedo Timoneda

Martin Töpfer

Tomáš Toufar

Lukáš Vik

We are grateful to DataRail for providing us free internet connection.

■ PANTONE 534C CMYK 99|82|35|27 RGB 29|52|92 HEX #1D345C

■ PANTONE 369C CMYK 67|8|100|0 RGB 98|168|48 HEX #62A830

DataRail®
DATA ON BOARD

Shannon '40s
→ OTP
→ inherent
Private Information Retrieval
(PIR)
Secure Compu
→ is it inherent
→ Computational
→ they exist
on CP