Algorithms on Strings

Michael Soltys

CSU Channel Islands
Computer Science

April 9, 2018

Basics

An *alphabet* is a finite, non-empty set of distinct symbols, denoted usually by Σ.

e.g., $\Sigma = \{0, 1\}$ (binary alphabet)
$\Sigma = \{a, b, c, \ldots, z\}$ (lower-case letters alphabet)

A *string*, also called *word*, is a finite ordered sequence of symbols chosen from some alphabet.

e.g., 010011101011

$|w|$ denotes the *length* of the string w.

e.g., $|010011101011| = 12$

The *empty string*, ε, $|\varepsilon| = 0$, is in any Σ by default.

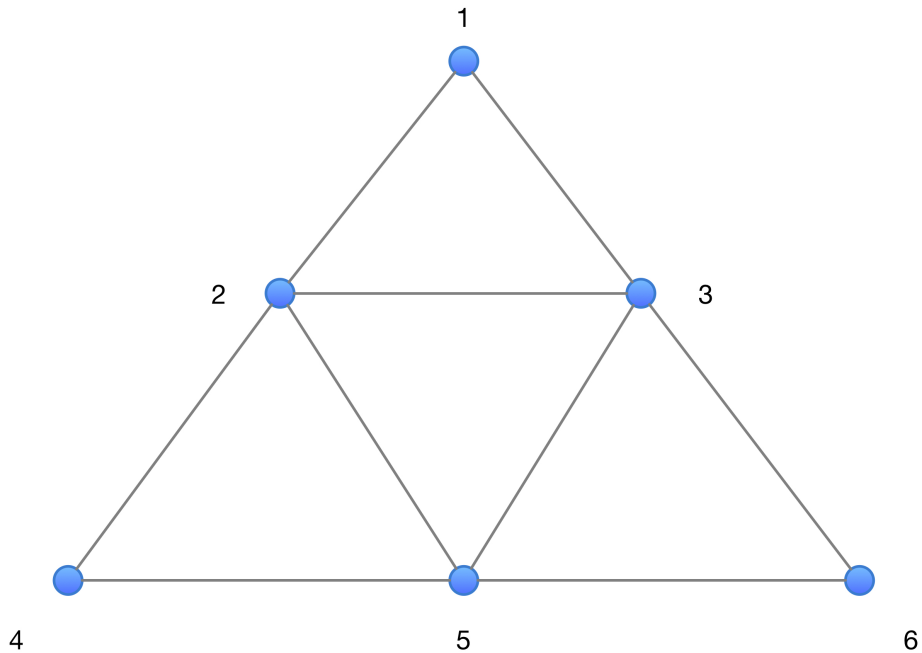
In this talk we discuss three papers and related open problems:

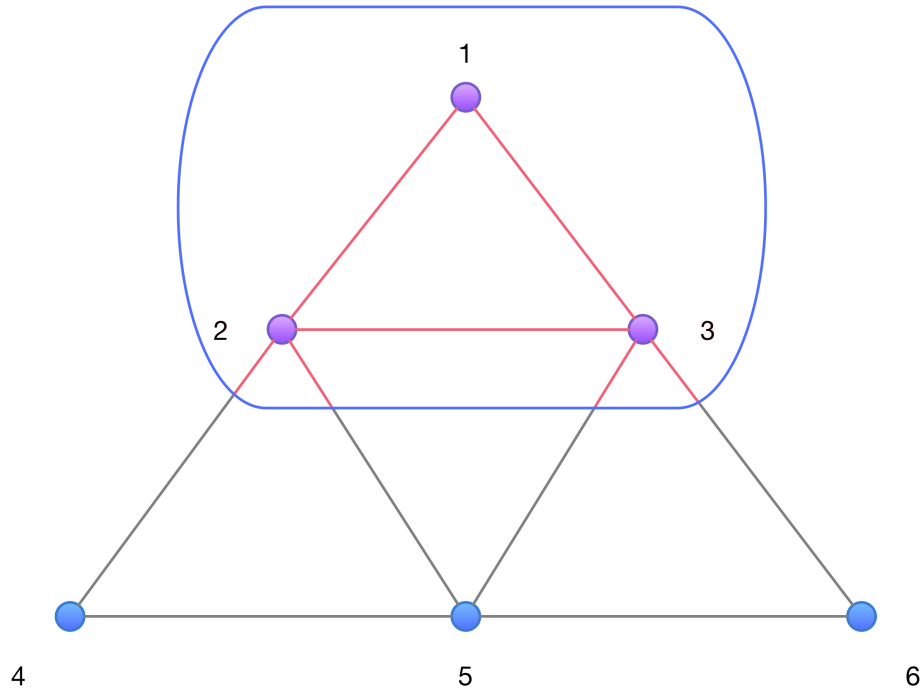
- Ryan McIntyre and Michael Soltys
  An improved upper bound and algorithm for clique covers
  Journal of Discrete Algorithms, 2018
  `https://doi.org/10.1016/j.jda.2018.03.002`
  `https://authors.elsevier.com/a/1Wo5K5bB7ekzdT`
- Michael Soltys and Neerja Mhaskar
  A formal framework for stringology
  Journal of Discrete Applied Mathematics, 2018
  `https://doi.org/10.1016/j.dam.2018.03.010`
- Sam Buss and Michael Soltys
  Unshuffling a square is NP-hard
  Journal of Computer and System Sciences, 2014
  `https://doi.org/10.1016/j.jcss.2013.11.002`

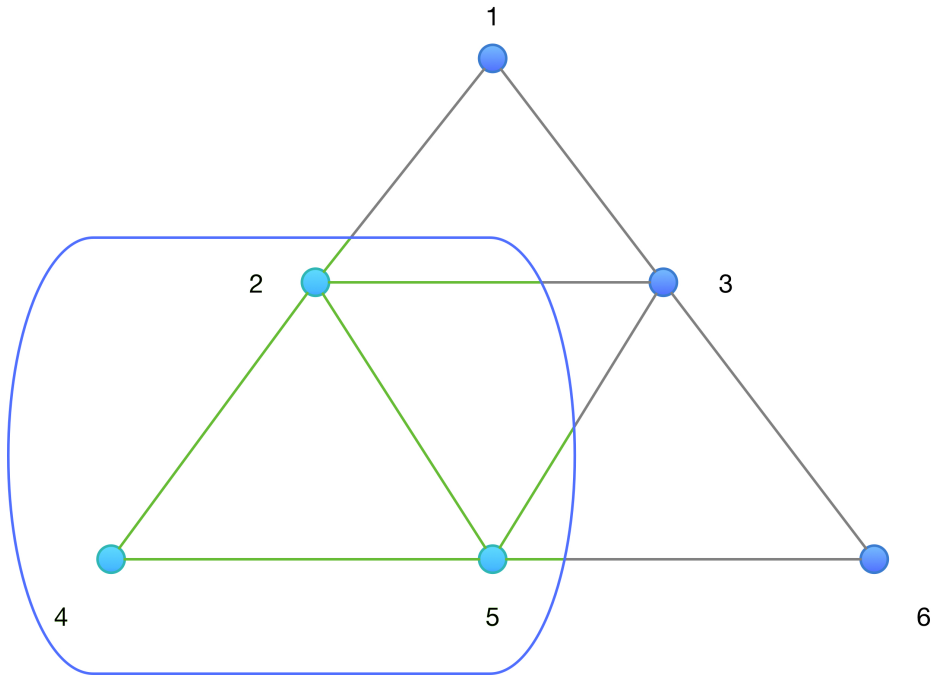
History of clique covers

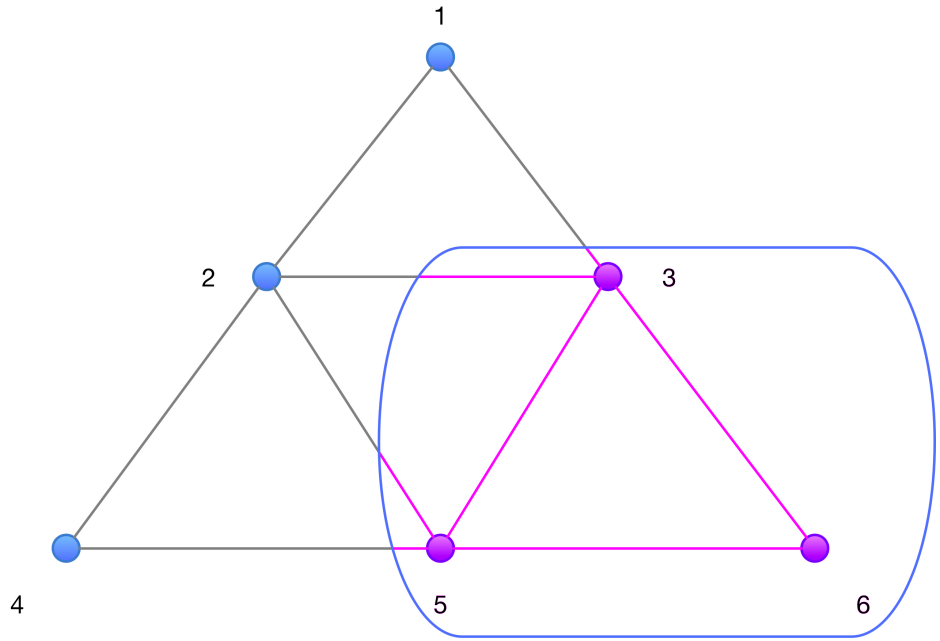
http://soltys.cs.csuci.edu/blog/?p=1963

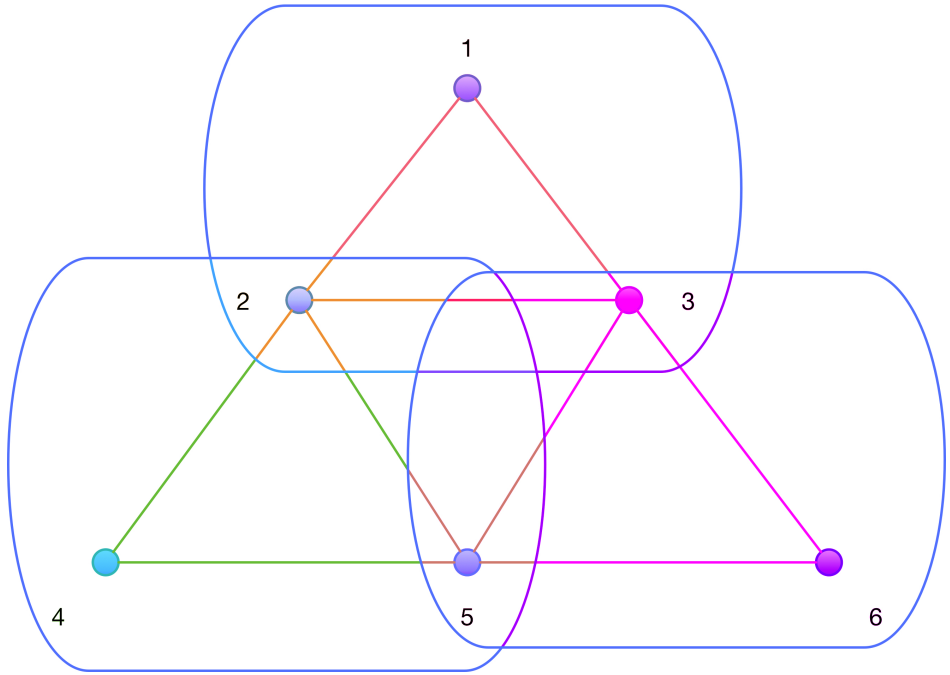
https://doi.org/10.1016/j.tcs.2017.02.016


1
2
3
4
5
6


1
2
3
4
5
6


1
2
3
4
5
6


1
2
3
4
5
6


1
2
3
4
5
6

Definitions

Given an undirected graph $G = (V, E)$, we say that $c \subseteq V$ is a *clique* if every pair of distinct vertices $(u, v) \in c \times c$ comprises an edge—that is, $(u, v) \in E$.

A vertex u is *covered* by c if $u \in c$. Similarly, edge (u, v) is covered by c if $\{u, v\} \subseteq c$; we will often write $(u, v) \in c$ instead.

$C = \{c_1, c_2, \cdots, c_k\}$ is a *clique cover* of G if size k if each c_i is a clique, and furthermore every edge and vertex in G is covered by at least one such c_i.

Given two integers n and m such that $n > 0$ and $0 \leq m \leq \binom{n}{2}$, we let $\mathcal{G}_{n,m}$ denote the set of all simple, undirected graphs on n vertices and m edges.

Given any graph G, we denote by $\theta(G)$ the size of a smallest cover of G.

Finally, we denote by $\Theta_n(m)$ the largest $\theta(G)$ of all graphs $G \in \mathcal{G}_{n,m}$.

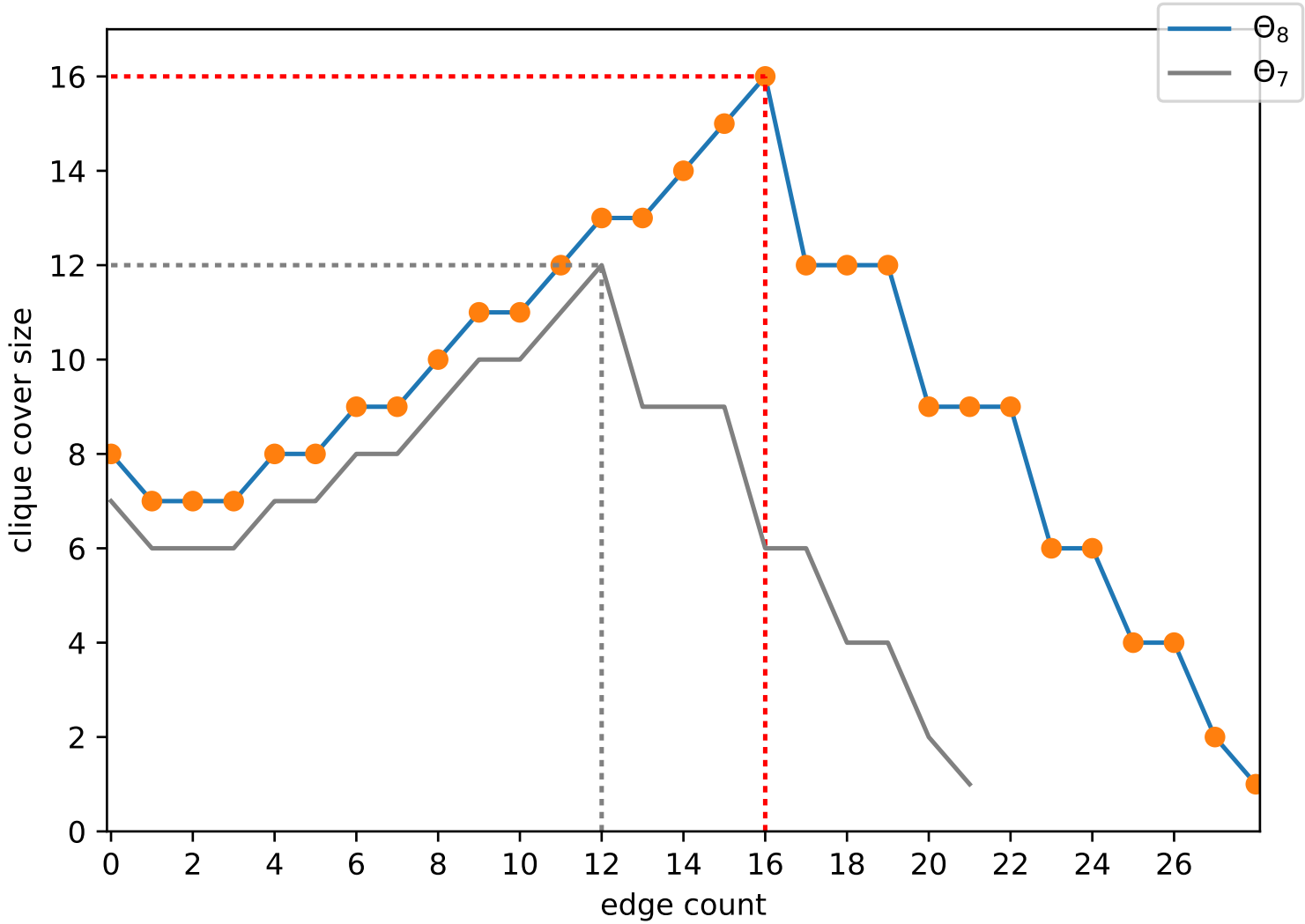
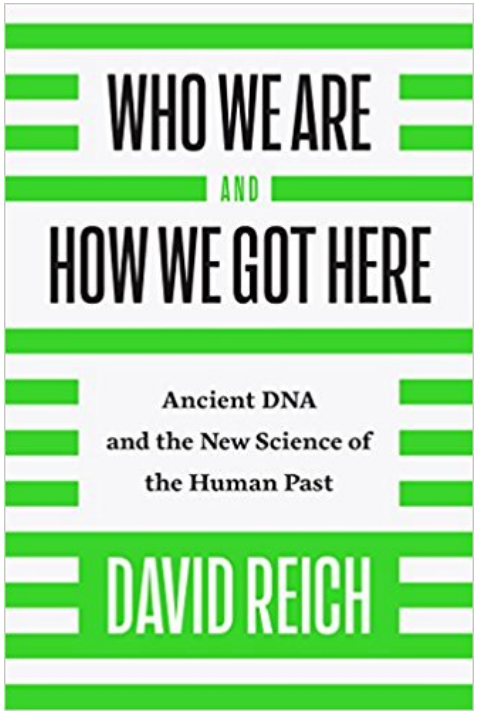
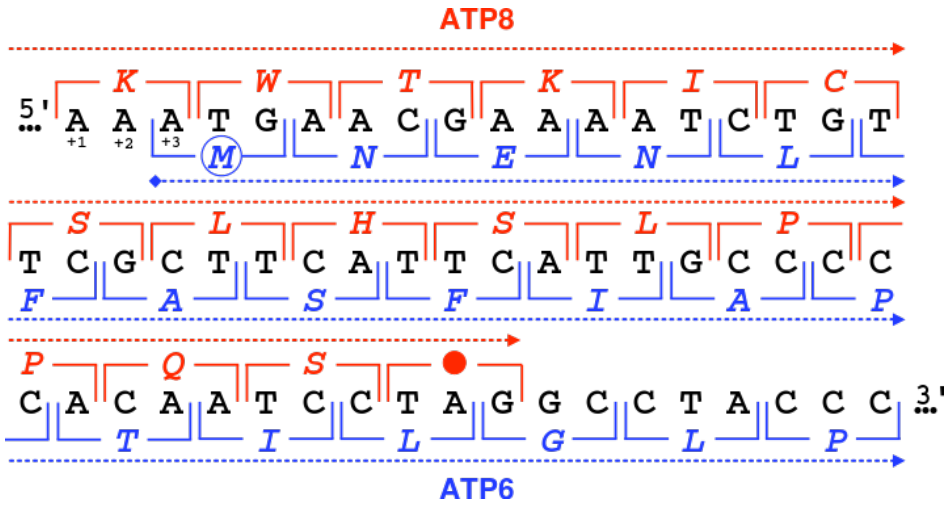


Figure: $\Theta_8(m)$ and $\Theta_7(m)$

Why are we interested in this?

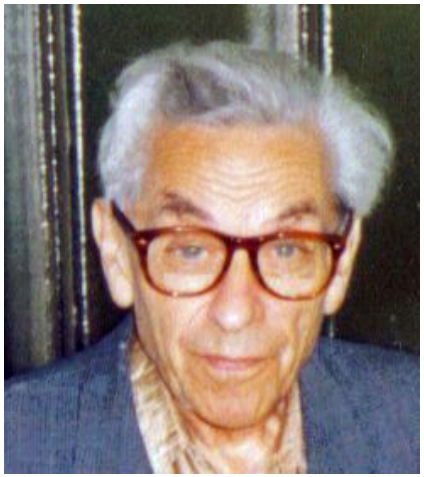




https://ghr.nlm.nih.gov/gene/MT-ATP6

CA{C,A,T}{G,A}TG{A,C}C{T,G,A}AACT

Erdősz, Lóvasz, Mantel



Paul Erdősz
1913–1996

László Lovász
1948–

“Keyser Sőze”
1907–?

We know from Helling et al, and from the results of Mantel and Erdős, that the global maximum of $\Theta_n(m)$ is reached at $m = \lfloor n^2/4 \rfloor$.

The reason is that this is the largest number of edges which can fit on n vertices without forcing triangles.

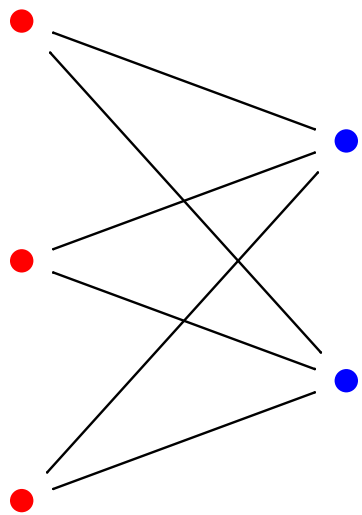
This maximum is realized in complete bipartite graphs—such graphs have no triangles or singletons, so covers consist of all edges.

The expression '$\lfloor n^2/4 \rfloor$' will be used frequently, so we abbreviate it: for any expression exp, we let $\overline{\text{exp}} = \lfloor \text{exp}^2/4 \rfloor$.

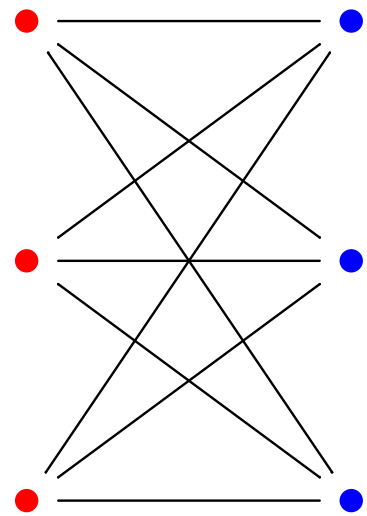
Next figure displays the largest complete bipartite graphs on five and six vertices respectively: $\mathcal{K}_{3,2}$ and $\mathcal{K}_{3,3}$.

Note that $\theta(\mathcal{K}_{3,2}) = 6 = \overline{5}$ and $\theta(\mathcal{K}_{3,3}) = 9 = \overline{6}$.

For any natural n, $\theta(\mathcal{K}_{\lceil n/2 \rceil, \lfloor n/2 \rfloor}) = \overline{n}$.



$\mathcal{K}_{3,2}$



$\mathcal{K}_{3,3}$

Results we build on

Theorem (Mantel, Erdős; *Thm 2 in paper*)

If a graph on n vertices contains no triangle, then it contains at most $\overline{n}$ edges.

Theorem (Lovász; *Thm 3 in paper*)

Given $G \in \mathcal{G}_{n,m}$, let k be the number of missing edges (i.e. $k = \binom{n}{2} - m$), and let t be the largest natural number such that $t^2 - t \leq k$. Then $\theta(G) \leq k + t$. Moreover, this bound is exact if $k = t^2$ or $k = t^2 - t$.

For $m \leq \overline{n}$, we rely primarily on the theorems above, provided by Mantel and Erdős; we use them to prove our first contribution, namely that $\Theta_n(m)$ has some recursive properties.

These properties provide an exact upper bound when $m \leq \overline{n}$.

Lovász provides an inexact upper bound when $m \geq \overline{n}$.

We have two improvements to Lovász's bound stated as Theorems below; these improvements comprise our most notable theoretical results in this paper.

We also give a conjecture; if proven true, this conjecture finishes the complete exact upper bound of for $m \geq \overline{n}$.

Theorem (*Thm 12 in paper)*

If $m > \overline{n}$ *then* $\Theta_n(m) \leq \overline{n-1}$.

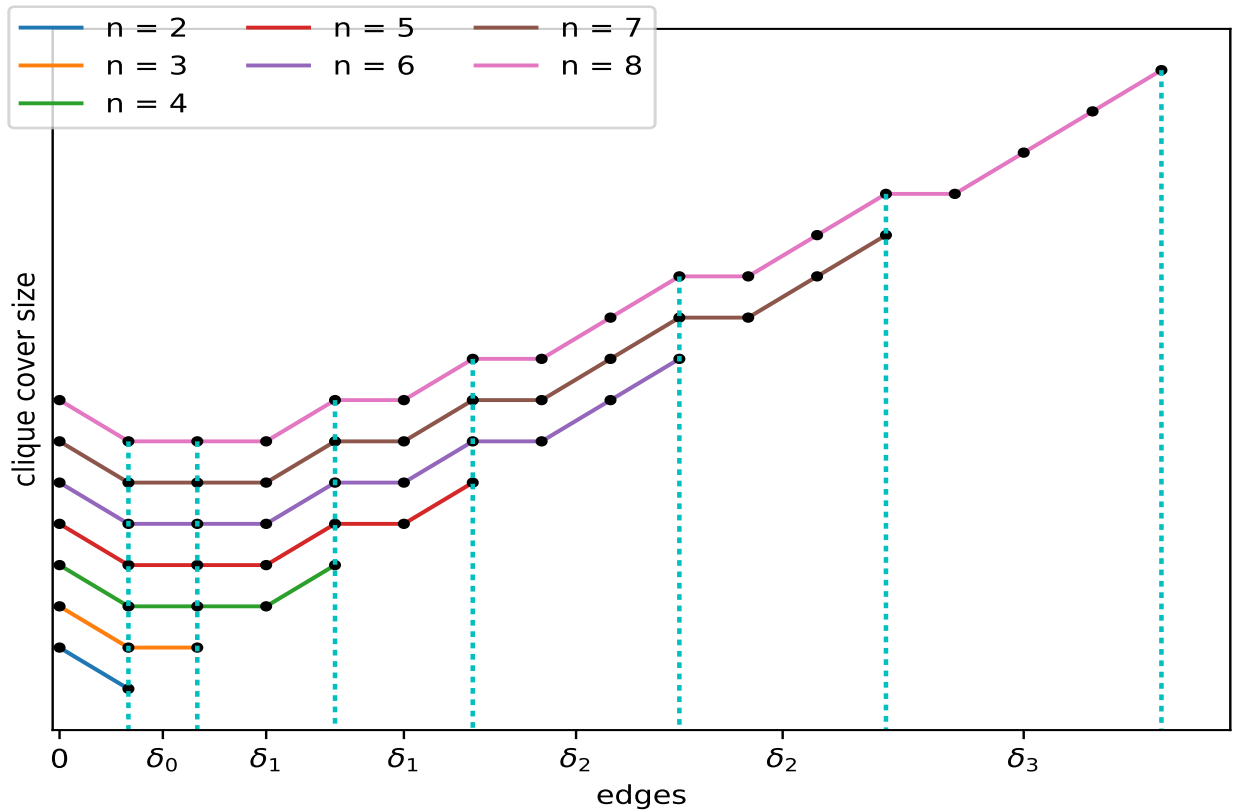
Theorem (*Thm 17 in paper)*

If $m > \binom{n}{2} - \overline{n-2}$ *then* $\Theta_n(m) \leq \overline{n-2}$.

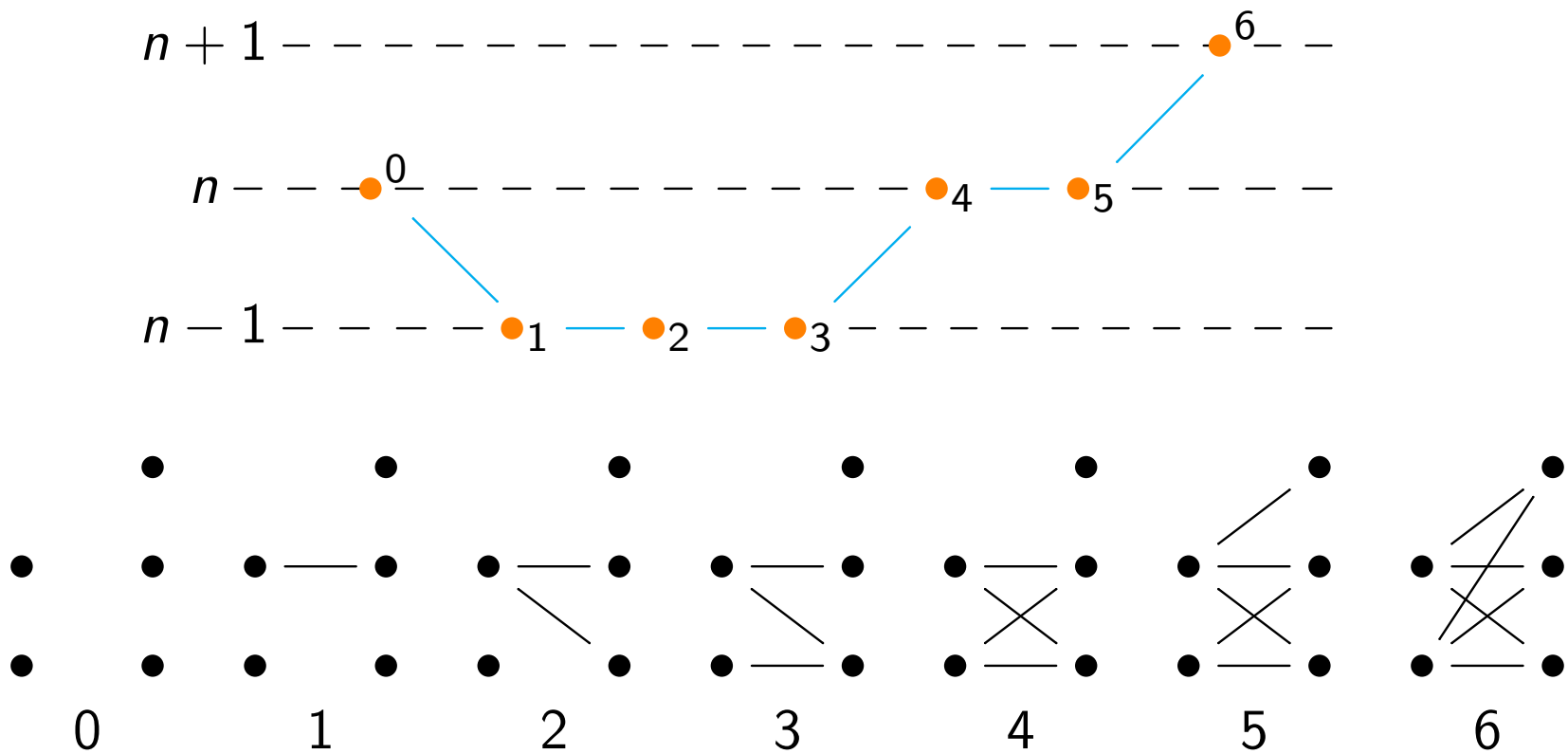
Conjecture (*Conjecture 14 in paper)*

If $k < \overline{p}$, *then* $\Theta_n(\binom{n}{2} - k) \leq \overline{p}$.

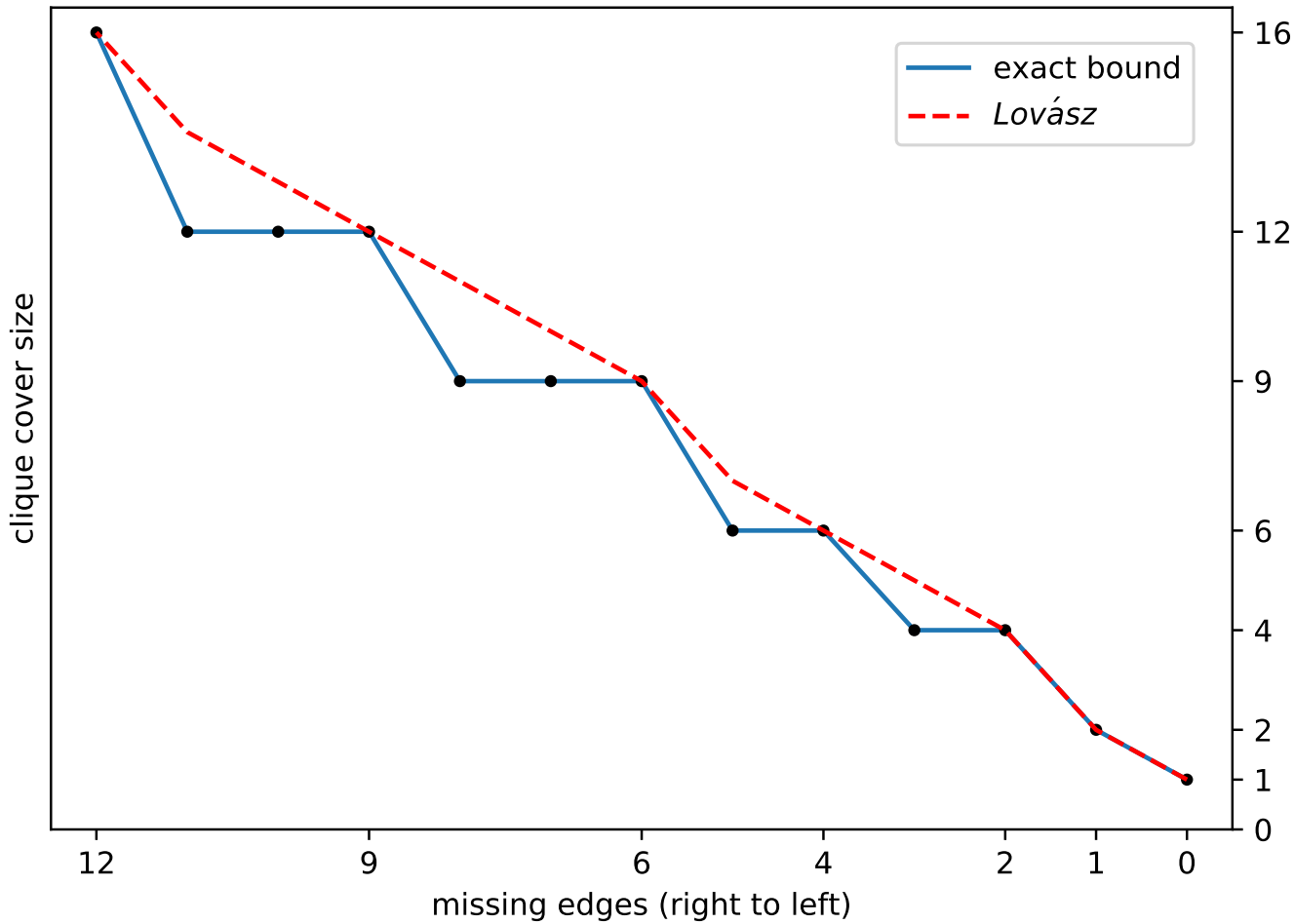
$\Theta_n(m)$ for $m \leq \overline{n}$



Visual justification for the first seven points of $\Theta_n(m)$ for $n \geq 5$.



$\Theta_n(m)$ for $m > \overline{n}$



Lovász's bound vs the right side of Θ_8

With Lovász's Theorem and a lemma, we can form an upper bound for Θ_n:

$$\Theta_n^{(2)}(m) \begin{cases} = \Theta_{n-1}(m) + 1 & \text{for } m \leq \overline{n-1} \quad (1a) \\ = m & \text{for } \overline{n-1} < m \leq \overline{n} \quad (1b) \\ \leq k + \max\{t | t^2 - t \leq k\} & \text{for } \overline{n} < m \leq \binom{n}{2} \quad (1c) \end{cases}$$

With our Theorems, we can improve the previous bound:

$$\Theta_n^{(3)}(m) \begin{cases} = \Theta_{n-1}(m) + 1 & \text{for } m \leq \overline{n-1} \quad (2a) \\ = m & \text{for } \overline{n-1} < m \leq \overline{n} \quad (2b) \\ = \overline{n-1} & \text{for } \overline{n-1} > k \geq \overline{n-2} \quad (2c) \\ = \overline{n-2} & \text{for } \overline{n-2} > k \geq \overline{n-3} \quad (2d) \\ \leq k + \max\{t | t^2 - t \leq k\} & \text{for } \overline{n-3} > k \geq 0 \quad (2e) \end{cases}$$

Open Problem

If conjecture is proven true, the bound can be simplified and made exact for all m:

$$\Theta_n^{(4)}(m) = \begin{cases} \Theta_{n-1}(m) + 1 & \text{for } m \leq \overline{n-1} \quad (3a) \\ m & \text{for } \overline{n-1} < m \leq \overline{n} \quad (3b) \\ \overline{\min\{t | \overline{t} > k\}} & \text{for } m > \overline{n} \quad (3c) \end{cases}$$

Note that these three formulations form a refinement of the upper bound; $\Theta_n^{(4)}(m) \leq \Theta_n^{(3)}(m) \leq \Theta_n^{(2)}(m)$ for all m. Conjecture is sufficient to show that $\Theta_n = \Theta_n^{(4)}$.

Ryan McIntyre will give a detailed talk about this result tomorrow:

7:30pm
April 10, 2018
Bell Tower 1462

This is the usual COMP/MATH 554 (Graduate Algorithms), but I am away attending an ABET accreditation workshop.

Proof Complexity

http://repository.cmu.edu/cgi/viewcontent.cgi?article=1923&context=compsci

"On the Unusual Effectiveness of Logic in Computer Science"

- A new formal framework for Stringology is proposed, which consists of a three-sorted logical theory S designed to capture the combinatorial reasoning about finite words.
- A witnessing theorem is proven which demonstrates how to extract algorithms for constructing strings from their proofs of existence.
- Various other applications of the theory are shown.
- The long term goal of this line of research is to introduce the tools of Proof Complexity to the analysis of strings.

Language

Formal	Informal	Intended Meaning
		Index
0_{index}	0	the integer zero
1_{index}	1	the integer one
$+_{\text{index}}$	$+$	integer addition
$-_{\text{index}}$	$-$	bounded integer subtraction
$\cdot_{\text{index}}$	$\cdot$	integer multiplication (we also just use juxtaposition)
$\text{div}_{\text{index}}$	div	integer division
$\text{rem}_{\text{index}}$	rem	remainder of integer division
$<_{\text{index}}$	$<$	less-than for integers
$=_{\text{index}}$	$=$	equality for integers
		Alphabet symbol
$\mathbf{0}_{\text{symbol}}$	$\mathbf{0}$	default symbol in every alphabet
σ_{symbol}	σ	unary function for generating more symbols
$<_{\text{symbol}}$	$<$	ordering of alphabet symbols
$\text{cond}_{\text{symbol}}$	cond	a conditional function
$=_{\text{symbol}}$	$=$	equality for alphabet symbols
		String
$\|\|_{\text{string}}$	$\|\|$	unary function for string length
e_{string}	e	binary fn. for extracting the i-th symbol from a string
$=_{\text{string}}$	$=$	string equality

λ string constructors

The string 000 can be represented by:

$$\lambda i \langle 1 + 1 + 1, \mathbf{0} \rangle.$$

Given an integer n, let $\hat{n}$ abbreviate the term $1 + 1 + \cdots + 1$ consisting of n many 1s. Using this convenient notation, a string of length 8 of alternating 1s and 0s can be represented by:

$$\lambda i \langle \hat{8}, \mathrm{cond}(\exists j \leq i(j + j = i), \mathbf{0}, \sigma\mathbf{0}) \rangle.$$

Let U be a binary string, and suppose that we want to define $\bar{U}$, which is U with every 0 (denoted $\mathbf{0}$) flipped to 1 (denote $\sigma\mathbf{0}$), and every 1 flipped to 0. We can define $\bar{U}$ as follows:

$$\bar{U} := \lambda i \langle |U|, \mathrm{cond}(e(U, i) = \mathbf{0}, \sigma\mathbf{0}, \mathbf{0} \rangle.$$

Index Axioms

Index Axioms	
B1. $i+1 \neq 0$	B9. $i \leq j, j \leq i \rightarrow i = j$
B2. $i+1 = j+1 \rightarrow i = j$	B10. $i \leq i+j$
B3. $i+0 = i$	B11. $0 \leq i$
B4. $i+(j+1) = (i+j)+1$	B12. $i \leq j \vee j \leq i$
B5. $i \cdot 0 = 0$	B13. $i \leq j \leftrightarrow i < j+1$
B6. $i \cdot (j+1) = (i \cdot j)+i$	B14. $i \neq 0 \rightarrow \exists j \leq i(j+1 = i)$
B7. $i \leq j, i+k = j \rightarrow j-i = k$	B15. $i \not\leq j \rightarrow j-i = 0$
B8. $j \neq 0 \rightarrow \mathrm{rem}(i,j) < j$	B16. $j \neq 0 \rightarrow i = j \cdot \mathrm{div}(i,j) + \mathrm{rem}(i,j)$

Symbol and String Axioms

Alphabet Axioms
B17. $u \lneqq \sigma u$
B18. $u < v, v < w \rightarrow u < w$
B19. $\alpha \rightarrow \mathrm{cond}(\alpha, u, v) = u$
B20. $\neg\alpha \rightarrow \mathrm{cond}(\alpha, u, v) = v$

String Axioms
B21. $\lvert\lambda i\langle t, s\rangle\rvert = t$
B22. $j < t \rightarrow e(\lambda i\langle t, s\rangle, j) = s(j/i)$
B23. $\lvert U\rvert \leq j \rightarrow e(U, j) = \mathbf{0}$
B24. $\lvert U\rvert = \lvert V\rvert, \forall i < \lvert U\rvert e(U, i) = e(V, i) \rightarrow U = V$

Conclusion

- If we can prove $\forall X \exists Y \alpha(X, Y)$, then we can compute the Y in polynomial time. (*Witnessing Theorem*.) So we can extract algorithms from proofs!
- Utilize the sophisticated tools of Proof Complexity for a combinatorial analysis of strings.

  the richness of the field of Stringology arises from the fact that a string U is a map $I \longrightarrow \Sigma$, where I can be arbitrarily large, while Σ is "small." This produces repetitions and patterns that are the object of study for Stringology. On the other hand, Proof Complexity has studied in depth the varied versions of the Pigeonhole Principle that is responsible for these repetitions.

Conclusion Cont'd

- The formalization allows us to see explicitly what is the engine of reasoning behind combinatorics on words.
  The Alphabet and String Axioms are *definitional*; they state the definitions of the objects.
  However, the Axioms for Indices provide the reasoning power.
  They show that combinatorics on words uses number theory on indices in order to prove its results.

Shuffle

w is the shuffle of u, v: $w = u \odot v$

$w = 0110110011101000$

$u = 01101110$

$v = 10101000$

$w = 0110110011101000$

Shuffle

w is the shuffle of u, v: $w = u \odot v$

$w = 0110110011101000$

$u = 01101110$

$v = 10101000$

$w = 0110110011101000$

w is a *shuffle* of u and v provided:

$u = x_1 x_2 \cdots x_k$

$v = y_1 y_2 \cdots y_k$

and w obtained by "interleaving" $w = x_1 y_1 x_2 y_2 \cdots x_k y_k$.

Square Shuffle

w is a *square* provided it is equal to a shuffle of a u with itself, i.e., $\exists u$ s.t. $w = u \odot u$

The string $w = 0110110011101000$ is a square:

$$w = 0110110011101000$$

and

$$u = 01101100 = 01101100$$

Result and Open Problems

given an alphabet Σ, $|\Sigma| \geq 7$,

$\textsc{Square} = \{w : \exists u(w = u \odot u)\}$

is NP-complete.

Problem 1:

- What about $|\Sigma| = 2$ (for $|\Sigma| = 1$, $\textsc{Square}$ is just the set of even length strings). More generally, what is the threshold?
- What about if $|\Sigma| = \infty$ but each symbol cannot occur more often than, say, 6 times (if each symbol occurs at most 4 times, $\textsc{Square}$ can be reduced to 2-$\textsc{Sat}$ – see P. Austrin *Stack Exchange* post `http://bit.ly/WATco3`)