The Closest Substring problem with small distances

Dániel Marx

Humboldt-Universität zu Berlin

dmarx@informatik.hu-berlin.de

IEEE Symposium on Foundations of Computer Science,

October 23, 2005

The Closest Substring problem

CLOSEST SUBSTRING

Input: Binary strings $s_1, \ldots, s_k$, integers L and d

Find: — string s of length L (center string),

— a length L substring s'_i of s_i for every i

such that $d(s, s'_i) \leq d$ for every i

Applications: finding common genetic patterns, drug design.

Problem is NP-hard even in the special case $|s_i| = L$.

Small parameters

Problem can be solved in...

- $2^L \cdot O(n)$ time,
- $n^{O(d)}$ time,
- $n^{O(k)}$ time.

Small parameters

Problem can be solved in...

- $2^L \cdot O(n)$ time,
- $n^{O(d)}$ time,
- $n^{O(k)}$ time.

Main question: Is there are an $n^{O(1)}$ algorithm for fixed d and/or k?

Can be studied in the framework of parameterized complexity.

Parameterized complexity

Goal: restrict the exponential growth of the running time to one parameter of the input.

Finding a path of length k: Can be done in $O(2^k \cdot n^2)$	vs.	Finding a clique of size k: No $n^{o(k)}$ algorithm is known

Parameterized complexity

Goal: restrict the exponential growth of the running time to one parameter of the input.

Finding a path of length k: Can be done in $O(2^k \cdot n^2)$	vs.	Finding a clique of size k: No $n^{o(k)}$ algorithm is known

In a **parameterized problem**, every instance has a special part k called the **parameter.**

Definition: A parameterized problem is **fixed-parameter tractable (FPT)** with parameter k if there is an algorithm with running time $f(k) \cdot n^c$ where c is a fixed constant not depending on k.

Parameterized intractability

We expect that MAXIMUM INDEPENDENT SET is not fixed-parameter tractable, no $n^{o(k)}$ algorithm is known.

W[1]-complete $\approx$ “as hard as MAXIMUM INDEPENDENT SET”

Parameterized intractability

We expect that MAXIMUM INDEPENDENT SET is not fixed-parameter tractable, no $n^{o(k)}$ algorithm is known.

W[1]-complete $\approx$ "as hard as MAXIMUM INDEPENDENT SET"

Parameterized reductions:

L_1 is reducible to L_2, if there is a function f: $(x,k) \mapsto (x',k')$ such that

- $(x,k) \in L_1 \iff (x',k') \in L_2$,
- f can be computed in $f(k) \cdot |x|^c$ time,
- **k' depends only on k**

If L_1 is reducible to L_2, and L_2 is in FPT, then L_1 is in FPT as well.

Closest Substring—Results

Fact: [Fellows et al. 2002] Problem is W[1]-hard with parameter k
$\Rightarrow$ no $f(k) \cdot n^{O(1)}$ algorithm (unless W[1]=FPT).

Closest Substring—Results

Fact: [Fellows et al. 2002] Problem is W[1]-hard with parameter k
$\Rightarrow$ no $f(k) \cdot n^{O(1)}$ algorithm (unless W[1]=FPT).

New results:

- Problem is W[1]-hard with combined parameters d and k
  $\Rightarrow$ no $f(k,d) \cdot n^{O(1)}$ time algorithm (unless W[1]=FPT).
- No $f(k,d) \cdot n^{o(\log d)}$ or $f(k,d) \cdot n^{o(\log \log k)}$ algorithm (unless n-variable 3-SAT can be solved in $2^{o(n)}$ time).
- Problem can be solved in $f(k,d) \cdot n^{O(\log d)}$ time.
- Problem can be solved in $f(k,d) \cdot n^{O(\log \log k)}$ time.

Hardness of Closest Substring

Theorem: Closest Substring is W[1]-hard with combined parameters k, d.

Proof by parameterized reduction from Maximum Independent Set.

Maximum Independent Set		Closest Substring
(G, t)	$\Rightarrow$	$k = 2^{2^{O(t)}}$ $d = 2^{O(t)}$

Corollary: No $f(k, d) \cdot n^{O(1)}$ algorithm for Closest Substring unless FPT=W[1].

Hardness of Closest Substring

Theorem: CLOSEST SUBTRING is W[1]-hard with combined parameters k, d.

Proof by parameterized reduction from MAXIMUM INDEPENDENT SET.

MAXIMUM INDEPENDENT SET (G, t)	$\Rightarrow$	CLOSEST SUBSTRING $k = 2^{2^{O(t)}}$ $d = 2^{O(t)}$

Corollary: No $f(k, d) \cdot n^{O(1)}$ algorithm for CLOSEST SUBSTRING unless FPT=W[1].

Corollary: No $f(k, d) \cdot n^{o(\log d)}$ or $f(k, d) \cdot n^{o(\log \log k)}$ algorithm unless MAXIMUM INDEPENDENT SET has an $f(t) \cdot n^{o(t)}$ algorithm.

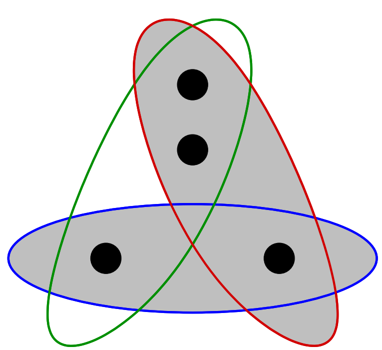
$\Rightarrow$ No such algorithm unless n-variable 3-SAT can be solved in $2^{o(n)}$ time.

(Fractional) edge covering

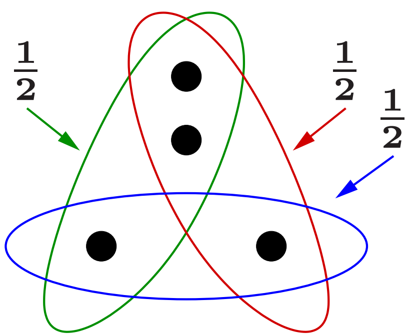
Hypergraph: each edge is an arbitrary set of vertices.

An **edge cover** is a subset of the edges such that every vertex is covered by at least one edge.
$\varrho(H)$: size of the smallest edge cover.

A **fractional edge cover** is a weight assignment to the edges such that every vertex is covered by total weight at least 1.
$\varrho^*(H)$: smallest total weight of a fractional edge cover.



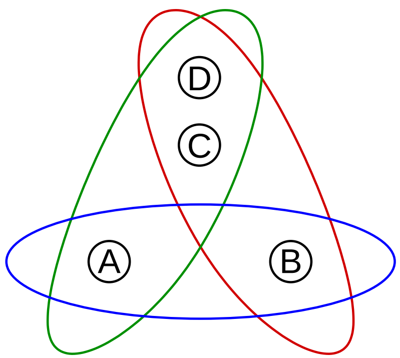
$\varrho(H) = 2$



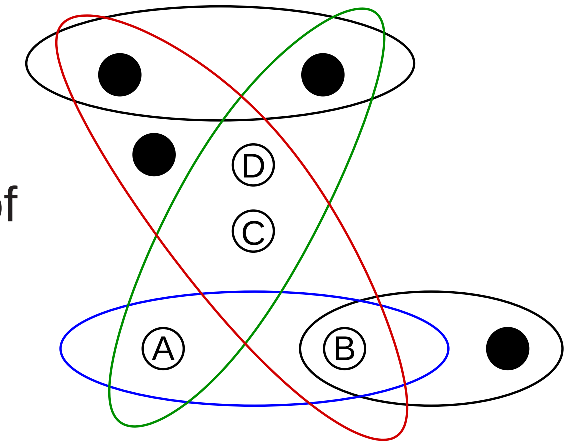
$\varrho^*(H) = 1.5$

Finding subhypergraphs

Subhypergraph: removing edges and vertices.

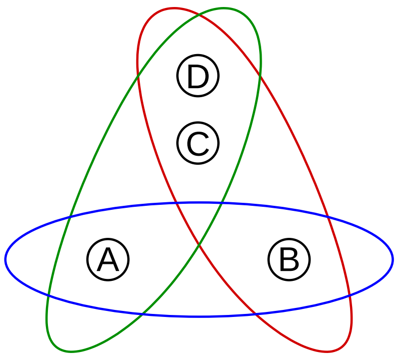


is a subhypergraph of

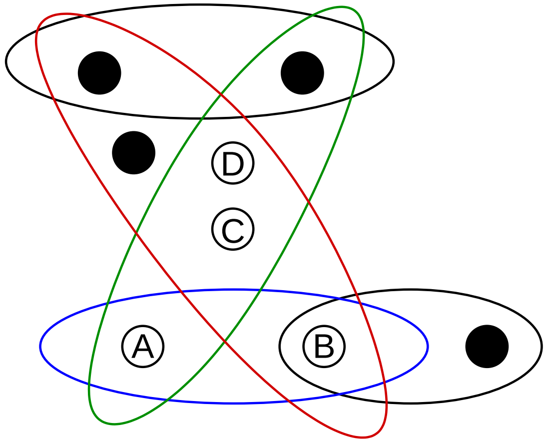


Finding subhypergraphs

Subhypergraph: removing edges and vertices.



is a subhypergraph of



We would like to enumerate all the places where H_1 appears in H_2.

Assuming that H_2 has m edges and each has size at most ℓ:

Lemma: [follows from Friedgut and Kahn 1998] H_1 can appear in H_2 at max. $f(\ell, \varrho^*(H_1)) \cdot m^{\varrho^*(H_1)}$ places.

Lemma: We can enumerate in $f(\ell, \varrho^*(H_1)) \cdot m^{O(\varrho^*(H_1))}$ time all the places where H_1 appears in H_2.

Half-covering

Defintion: A hypergraph has the half-covering property if for every non-empty set X of vertices there is an edge Y with $|X \cap Y| > |X|/2$.

Lemma: If a hypergraph H with m edges has the half-covering property, then $\varrho^*(H) = O(\log\log m)$.

Proof: by probabilistic arguments.

(The $O(\log\log m)$ is best possible.)

Reminder

Closest Substring

Input:	Binary strings $s_1, \ldots, s_k$, integers L and d
Find:	— string s of length L (center string),
	— a length L substring s'_i of s_i for every i
	such that $d(s, s'_i) \leq d$ for every i

The $f(k,d) \cdot n^{O(\log\log k)}$ algorithm

First step: guess the correct s'_1 ($\leq n$ possibilities).

Consider the set $\mathcal{S}$ of all length L substrings of $s_1, \ldots, s_k$. We turn $\mathcal{S}$ into a hypergraph H on vertices $\{1, 2, \ldots, L\}$: if a string in $\mathcal{S}$ differs from s'_1 on positions $P \subseteq \{1, 2, \ldots, L\}$, then let P be an edge of H.

The $f(k,d) \cdot n^{O(\log\log k)}$ algorithm

First step: guess the correct s'_1 ($\leq n$ possibilities).

Consider the set $\mathcal{S}$ of all length L substrings of $s_1, \dots, s_k$. We turn $\mathcal{S}$ into a hypergraph H on vertices $\{1, 2, \dots, L\}$: if a string in $\mathcal{S}$ differs from s'_1 on positions $P \subseteq \{1, 2, \dots, L\}$, then let P be an edge of H.

Lemma: Assume that in a solution s differs from s'_1 on positions P, and $d(s, s'_1)$ is as small as possible.
Then there is a hypergraph H_0 with at most d vertices and k edges having the half-covering property such that H_0 appears at P in H.

Algorithm: Consider every hypergraph H_0 as above and enumerate all the places where H_0 appears in H.

The $f(k,d) \cdot n^{O(\log\log k)}$ algorithm (cont.)

Algorithm:

- Guess s'_1.
- Construct the hypergraph H.
- Enumerate every hypergraph H_0 with at most d vertices and k edges (constant number).
- Check if H_0 has the half-covering property.
- If so, then enumerate every place P where H_0 appears in H.
(max. $\approx n^{O(\varrho^*(H_0))} = n^{O(\log\log k)}$ places).
- For each place P, check if there is a good center string that differs from s'_1 only at P.

Running time: $f(k,d) \cdot n^{O(\log\log k)}$.

Conclusions

- Parameterized analysis of CLOSEST SUBSTRING.
- Tight bounds on the exponent of n.
- Other applications of finding hypergraphs with small fractional edge cover number?