The Longest Run Subsequence Problem: Further Complexity Results

Riccardo Dondi, **Florian Sikora**

Università degli Studi di Bergamo, Bergamo, Italy,
LAMSADE, Université Paris Dauphine, CNRS – France

CPM - 07/21

The Longest Run Subsequence Problem: Further Complexity Results

Riccardo Dondi, **Florian Sikora**

Università degli Studi di Bergamo, Bergamo, Italy,
LAMSADE, Université Paris Dauphine, CNRS – France

Outline

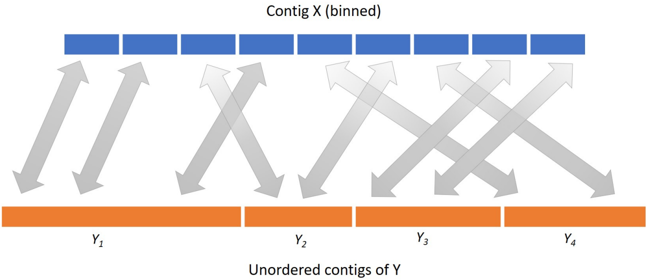
Story

- September 2020, we could attend WABI online
- We were independently interested by one open problem during a talk [Schrinner et al.] :

Is Longest Run Subsequence FPT w.r.t. the “compressed” size of the solution?

Motivations - reconstruct a genome

- Assembly from a set of reads to contigs
- Need to **order** them (scaffolding)
- Some matches are known (according to similarities)
- Some inconsistencies (errors in sequencing, mutations..)



$$S = y_1 y_1 y_2 y_1 y_4 y_2 y_4 y_3 y_3$$

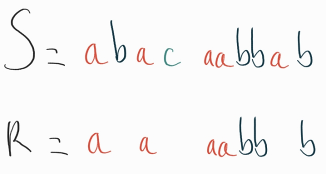
$$R = y_1 y_1 \quad y_1 y_4 \quad y_4 y_3 y_3$$

Definitions [Schrinner et al.]

- Given a string S over an alphabet Σ
- Find the longest subsequence R s.t. each symbol occurs only consecutively (or none)

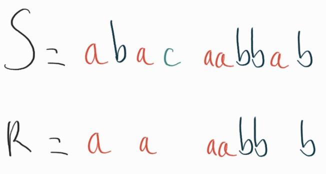
Definitions [Schrinner et al.]

- Given a string S over an alphabet Σ
- Find the longest subsequence R s.t. each symbol occurs only consecutively (or none)



Definitions [Schrinner et al.]

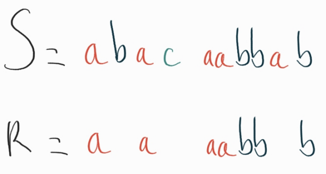
- Given a string S over an alphabet Σ
- Find the longest subsequence R s.t. each symbol occurs only consecutively (or none)



- An a-run : substring repeating symbol a
- Could have no maximal run in the solution
- Some symbol may not be in the solution

Definitions [Schrinner et al.]

- Given a string S over an alphabet Σ
- Find the longest subsequence R s.t. each symbol occurs only consecutively (or none)



- An a-run : substring repeating symbol a
- Could have no maximal run in the solution
- Some symbol may not be in the solution
- Interesting parameters:
  - k : size of the solution
  - $|\Sigma|$: size of the alphabet
  - r : number of runs in the optimal solution

Known results [Schrinner et al.]

- NP-hard
- FPT w.r.t. $|\Sigma|$ (implemented)
- ILP

Known results [Schrinner et al.]

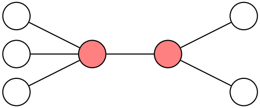
- NP-hard
- FPT w.r.t. $|\Sigma|$ (implemented)
- ILP

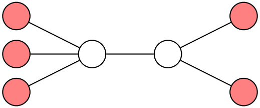
Open :

- FPT w.r.t. r ?
- Approximability ?

Parameterized Complexity

Problem:	Vertex Cover	Independent Set
Input:	Graph G, integer k	Graph G, integer k
Question:	Cover edges with k vertices	Find k independent vertices

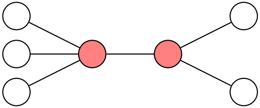
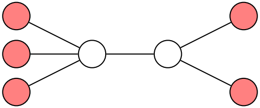




Compl.:

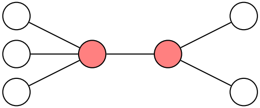
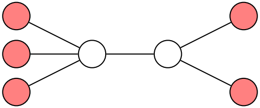
Example from D. Marx.

Parameterized Complexity

Problem:	Vertex Cover	Independent Set
Input:	Graph G, integer k	Graph G, integer k
Question:	Cover edges with k vertices	Find k independent vertices
Compl.:		
Complexity:	NP-complete	NP-complete

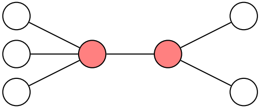
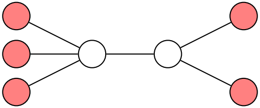
Example from D. Marx.

Parameterized Complexity

Problem:	Vertex Cover	Independent Set
Input:	Graph G, integer k	Graph G, integer k
Question:	Cover edges with k vertices	Find k independent vertices
Compl.:		
Complexity:	NP-complete	NP-complete
Brute-force:	$O(n^k)$ possibilities	$O(n^k)$ possibilities

Example from D. Marx.

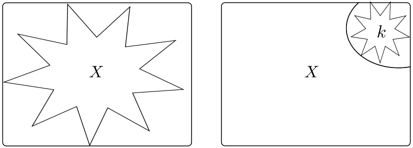
Parameterized Complexity

Problem:	Vertex Cover	Independent Set
Input:	Graph G, integer k	Graph G, integer k
Question:	Cover edges with k vertices	Find k independent vertices
		
Compl.:		
Complexity:	NP-complete	NP-complete
Brute-force:	$O(n^k)$ possibilities	$O(n^k)$ possibilities
Smarter?:	$O(2^k n^2)$ **algorithm**	**No** $f(k)n^{O(1)}$ **algorithm exists**
		

Example from D. Marx.

Fixed-Parameter Tractability

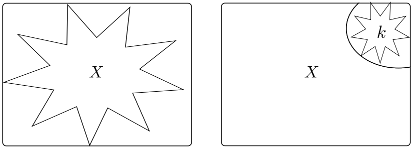
- Problem in FPT: any instance (I, k) solved in $f(k) \cdot |I|^c$.



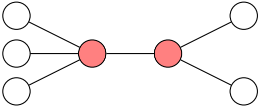
- Examples:
  - Solution of size k in a n-vertices graph.
  - n voters for k candidates.
  - Requests of size k in a n-sized database.
  - ...

Fixed-Parameter Tractability

- Problem in FPT: any instance (I, k) solved in $f(k) \cdot |I|^c$.

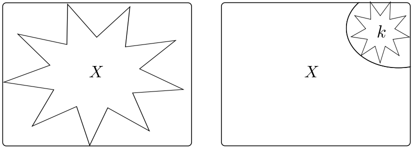


- Examples:
  - Solution of size k in a n-vertices graph.
  - n voters for k candidates.
  - Requests of size k in a n-sized database.
  - ...
- Many way to parameterize.
  - Solution size.



Fixed-Parameter Tractability

- Problem in FPT: any instance (I, k) solved in $f(k) \cdot |I|^c$.



- Examples:
  - Solution of size k in a n-vertices graph.
  - n voters for k candidates.
  - Requests of size k in a n-sized database.
  - ...
- Many way to parameterize.
  - Solution size.
  - Structure of the input.
  - ...

How to obtain FPT algorithm?

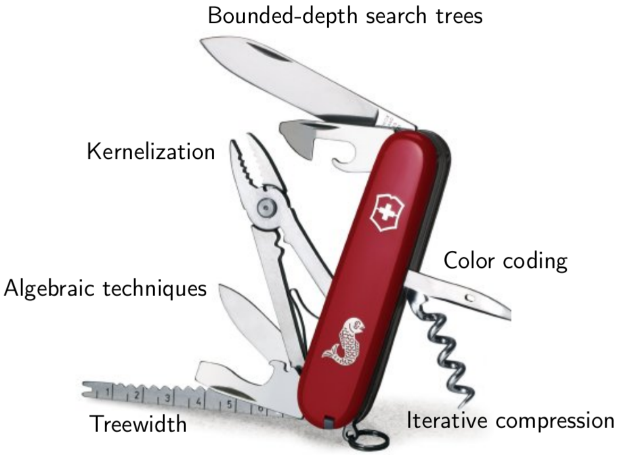
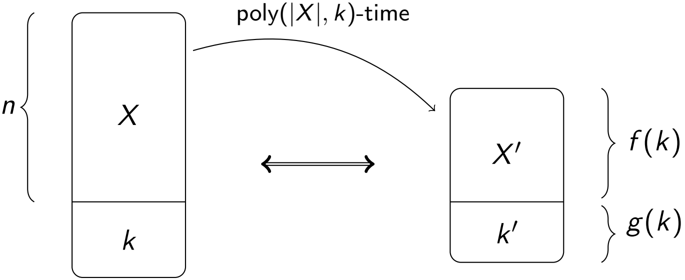


Illustration D. Marx.

Kernelization



Squirrel from [CFKLMPPS'15]

Parameterized Complexity or LRS

Parameters of decreasing size :

	FPT	Poly Kernel
k	Yes	**Yes**
$\|\Sigma\|$	Yes [Schrinner et al.]	**No**
r	**Yes & Poly Space**	**No**

$$S = abacaabbab$$

$|\Sigma| \leqslant k$: $R = abc$

$r \leqslant |\Sigma|$: $R = aaaabbb$

Outline

LRS and polynomials

- Color-Coding could probably work for parameter r
- We use a key result of [Koutis and Williams 2008,2009]:

Randomized algorithm to decide in **time $O^*(2^k)$ and polynomial space**, if a polynomial represented by an arithmetic circuit contains a **multilinear monomial of degree k**.

Polynomials

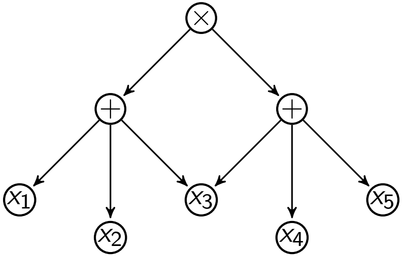
- A monomial is **multilinear** if each variable of the monomial occurs at most once.
- By definition, the degree of a multilinear monomial is the number of its variables.
- Example: $P(X) = (x_1^2 x_3 x_5 + x_1 x_2 x_4 x_6)$:
  - $x_1 x_2 x_4 x_6$ is a multilinear monomial of degree 4.
  - $x_1^2 x_3 x_5$ is not a multilinear monomial.

Compressed Polynomials

- An **arithmetic circuit** over a set of variables X is a DAG s.t.:
  - internal nodes are the operations $\times$ or $+$,
  - leafs are the elements of X.

Compressed Polynomials

- An **arithmetic circuit** over a set of variables X is a DAG s.t.:
  - internal nodes are the operations $\times$ or $+$,
  - leafs are the elements of X.
- Example for $P(X) = (x_1 + x_2 + x_3)(x_3 + x_4 + x_5)$.



Polynomials

- Framework successfully applied for different problems:
  - k-Path.
  - k-Tree.
  - k-Leaf Spanning Tree.
  - t-Dominating Set.
  - Graph Motif.
  - Exemplar Breakpoint Distance.
  - ...

To solve LRS w.r.t. r

- **A variable for each symbol** of the alphabet representing a potential run in the solution
- **Circuit** built via DP more or less representing subsequences
- A **multilinear** monomial of degree r in the circuit iff there is a solution with r runs

Outline

Size of the Kernel

- In FPT for $k \iff$ there is a kernel for k
- But $f(k)$ **could be anything** !
- Better if f is poly

Size of the Kernel

- In FPT for $k \iff$ there is a kernel for k
- But $f(k)$ **could be anything** !
- Better if f is poly

	FPT	Poly Kernel
k	Yes	**Yes**
$\|\Sigma\|$	Yes [Schrinner et al.]	**No**
r	**Yes & Poly Space**	**No**

Trivial poly-kernel for parameter k

- If there is one a-**run of size at least** k, done

$k = 3, S = aabca, R = aaa$

Trivial poly-kernel for parameter k

- If there is one a-**run of size at least** k, done
- If **alphabet is larger than** k, done

$k = 3, S = aabca, R = abc$

Trivial poly-kernel for parameter k

- If there is one a-**run of size at least** k, done
- If **alphabet is larger than** k, done
- Alphabet is at most k, no a-run is larger than k : **at most** k^2 **characters**

Hardness

- Informally, one way to prove no poly kernel :

Given t instances of our problem, build a “join” instance I' s.t. I' is true iff at least one of the t instance is true (OR-cross-composition)
[Bodlaender et al.]

Hardness

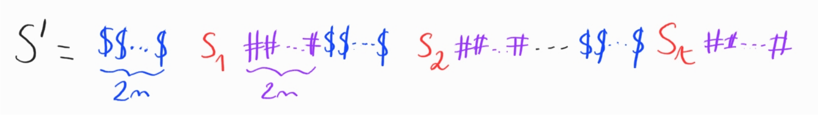
- Informally, one way to prove no poly kernel :

Given t instances of our problem, build a "join" instance I' s.t. I' is true iff at least one of the t instance is true (OR-cross-composition) [Bodlaender et al.]

- With technical details, we assume that the t instances:
  - have same size
  - have same k
  - are built over the same alphabet

Hardness

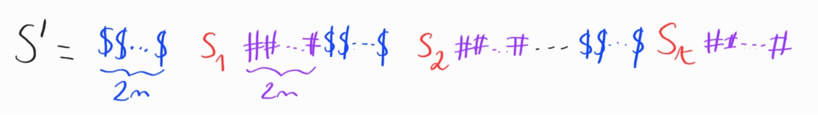
Add substrings of $2n$ new symbols around each instance



Look for a solution of size $(t+1) \times 2n + k$

Hardness

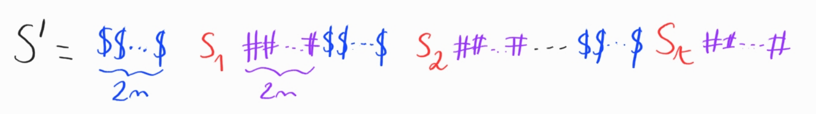
Add substrings of $2n$ new symbols around each instance



Look for a solution of size $(t + 1) \times 2n + k$
If there is a solution of size k in some S_i, take it and add all the \$ before and all the # after

Hardness

Add substrings of $2n$ new symbols around each instance



Look for a solution of size $(t+1) \times 2n + k$
If there is a solution of size k in some S_i, take it and add all the \$ before and all the # after

Parameter $|\Sigma|$ is the same plus 2

Outline

Complexity

- Better understanding of the parameterized complexity status of the problem

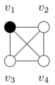
Complexity

- Better understanding of the parameterized complexity status of the problem
- Also, **LRS is APX-hard**, even with at most 2 occurrences for each symbol.
- (L-)Reduction from Independent Set in cubic graphs

Complexity

- Better understanding of the parameterized complexity status of the problem
- Also, **LRS is APX-hard**, even with at most 2 occurrences for each symbol.
- (L-)Reduction from Independent Set in cubic graphs

"No talk is complete without a picture of reduction that nobody understands" Daniel Marx



$$
\begin{aligned}
S = {} & w_1 x^1_{1,2} x^1_{1,3} x^1_{1,4} w_1 \sharp_{1,1} \sharp_{1,2} \sharp_{1,3} \\
& w_2 x^2_{1,2} x^2_{2,3} x^2_{2,4} w_2 \sharp_{2,1} \sharp_{2,2} \sharp_{2,3} \\
& w_3 x^3_{1,3} x^3_{2,3} x^3_{3,4} w_3 \sharp_{3,1} \sharp_{3,2} \sharp_{3,3} \\
& w_4 x^4_{1,4} x^4_{2,4} x^4_{3,4} w_4 \sharp_{4,1} \sharp_{4,2} \sharp_{4,3} \\
& e^1_{1,2} x^1_{1,2} e^2_{1,2} e^1_{1,2} x^2_{1,2} e^2_{1,2} \sharp_{5,1} \sharp_{5,2} \sharp_{5,3} \\
& e^1_{1,3} x^1_{1,3} e^2_{1,3} e^1_{1,3} x^3_{1,3} e^2_{1,3} \sharp_{6,1} \sharp_{6,2} \sharp_{6,3} \\
& e^1_{1,4} x^1_{1,4} e^2_{1,4} e^1_{1,4} x^4_{1,4} e^2_{1,4} \sharp_{7,1} \sharp_{7,2} \sharp_{7,3} \\
& e^1_{2,3} x^2_{2,3} e^2_{2,3} e^1_{2,3} x^3_{2,3} e^2_{2,3} \sharp_{8,1} \sharp_{8,2} \sharp_{8,3} \\
& e^1_{2,4} x^2_{2,4} e^2_{2,4} e^1_{2,4} x^4_{2,4} e^2_{2,4} \sharp_{9,1} \sharp_{9,2} \sharp_{9,3} \\
& e^1_{3,4} x^3_{3,4} e^2_{3,4} e^1_{3,4} x^4_{3,4} e^2_{3,4} \sharp_{10,1} \sharp_{10,2} \sharp_{10,3}
\end{aligned}
$$

Conclusion

- Practical issues of the FPT algorithm
- Close the gap of approximation (trivial $\sqrt{|S|}$-approx, far from the APX-hardness)

Dziękuję!