A Sketch of Parameterized Complexity

Hans Bodlaender

Utrecht University

Graph Drawing 2025

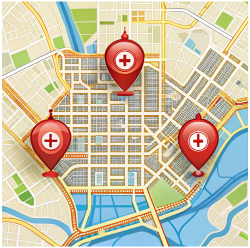
This talk

- An Introduction to Parameterized Complexity
  - To showcase important notions from the field
  - To give inspiration to use these ideas for the problems that *you* study
  - This talk has focus on complexity
  - Examples often from graph drawing

1. Overview of existing notions
2. New development: XNLP, XALP

Easy problems with small parameters

- Consider facility location problem:
  Place as few as possible fire stations in a city such that each house is $<$ 15 minutes drive from fire station.
- Problem is NP-hard, but ...
- Easy if we have just money for three fire stations: try all possible locations : $O(n^3)$.



Theory of Parameterized Complexity

- Many hard problems become easier / polynomial time solvable when a parameter is small/fixed.
- Early 1990s: Downey and Fellows build theory of parameterized complexity: what is the time complexity when we assume that some parameter k of the input or output is considered to be small?
- Subfield of algorithms research with:
  - New terminology
  - Conferences, workshops
  - Hundreds (thousands?) of papers . . .

Parameterized problem

- Parameterized problem: subset of $\Sigma^* \times \mathbb{N}$, with Σ a finite alphabet.
  - We call the second argument the parameter: usually denoted by k.
- Compare with 'classic' problem: subset of Σ^*.
- Time complexity of algorithm is $T(n,k)$
  (where in the classic setting we have $T(n)$)

Parameters

There are many different types of parametrizations possible:

- Target value
- Aspect of input size
- Structural parameter of input

Some examples: ...

Parameterization: target value

k-Planarity

Given: Graph G, integer k.
Parameter: k.
Question: can we embed G in the plane, such that no edge has more than k crossings?

Planar edge deletion

Given: Graph G, integer k.
Parameter: k.
Question: can we turn G into a planar graph by removing at most k edges?

Parameterization: input 'size'

Planar embedding with edge length constraints

Given: Planar graph $G = (V, E)$, for each edge $e \in E$, an interval $I_e \subseteq \mathbb{R}$.
Parameter: the number of vertices of G, $|V|$.
Question: is there a plane embedding of G such that for each edge e, the length of e in the embedding is a value in I_e?

Parameterization: structural value of input

Rectilinear planarity testing (treewidth)

Given: Planar graph G.
Parameter: the treewidth of G.
Question: Can we draw G such that each edge is drawn as a horizontal or vertical line segment?

Upwards Planarity Drawing (nb sources)

Given: Directed acyclic graph $G = (V, E)$.
Parameter: number d of vertices of indegree 0.
Question: Is there a plane drawing of G with for each arc, the y-coordinate increases?

Parameterization: combined parameters

k-Planarity(k+pw)

Given: Graph G, integer k
Parameter: $k + \mathrm{pathwidth}(G)$
Question: can we embed G in the plane, such that no edge has more than k crossings?

Theorem (Gima et al., GD 2025)

1-planarity is NP-complete for graphs of pathwidth 4.

Parameterized complexity

What is the time complexity of a problem as a function of both:

- the input size ($|V|$ or number of bits to write the input)
- and the value of the parameter?

'Classic' cases:

- Polynomial. Time of form $(n+k)^{O(1)}$
- Para-NP-complete: there is a value of the parameter k for which the problem with this parameter is NP-complete.

A para-NP-complete problem

As 1-Planarity testing is NP-complete for graphs of pathwidth 4 (Gima et al., GD 2025), k-Planarity is para-NP-complete for the combined parameter k and pathwidth.

For each value of k polynomial

Downey and Fellows, 1990s, define two flavours of problems that are polynomial for each value of k:

FPT (Fixed Parameter Tractable)

There is an algorithm that uses $f(k)n^{O(1)}$ time.

XP (Slice-wise polynomial time)

There is an algorithm that uses $n^{f(k)}$ time.

Different parameterizations — Different complexities

The same problem, but with different parameterisations, can have varying parameterized complexity

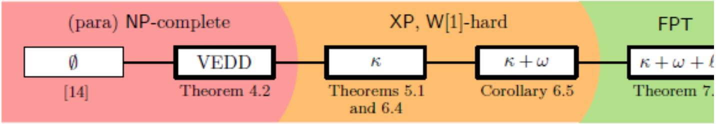


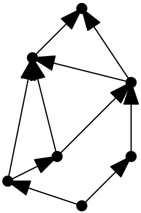
Figure 2 The complexity landscape of Stack Layout Extension. VEDD denotes the tex+edge deletion distance, ω denotes the page width of the ℓ-page stack layout of H, $\kappa = |V(G) \setminus V(H)| + |E(G) \setminus E(H)|$. Boxes outlined in bold represent new results that we sho the linked theorems and corollaries. The only result that is not depicted is Theorem 3.2.

Figure: From: Depian et al., Graph Drawing 2024: The Parameterized Complexity Of Extending Stack Layouts

Upwards Planarity

Upwards Planarity

Given: Directed acyclic graph G
Question: Can we draw G without crossing arcs, such that all arcs are increasing in the y-direction?



Theorem (Garg, Tamassia, 2001)

UPWARDS PLANARITY *is NP-complete.*

Results for Upwards Planarity (selected results)

Different parameterizations give different complexities:

- FPT by treedepth (Chaplick et al, 2022)
- FPT by number of sources (Chaplick et al, 2022)
- No polynomial kernel by treedepth (sketch later in this talk)
- XP by treewidth (Chaplick et al, 2022)
- $W[1]$-hard by treewidth (Jansen et al., GD 2023)
- XALP-complete by treewidth (corollary from J et al, BS)

FPT

FPT (Fixed Parameter Tractable) is the class of parameterized problems with an algorithm that runs in time $f(k)n^{O(1)}$, with f a computable function.

Remark

Variants:

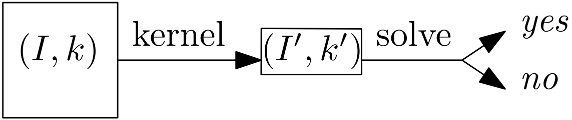
- weakly uniform FPT: f is not necessarily computable. In theory different, but no practical examples.
- non-uniform FPT: for each k, we can have another algorithm. Example: results from graph minor theory.

FPT and XP

- Complexity of XP is much higher than FPT: $f(k)n^{O(1)}$ vs $n^{f(k)}$.
- Relation with kernelisation(next).
- Downey-Fellows (1990s): Theory to show that problems are unlikely to be in FPT (W-hierarchy later in the talk).
- Proved with diagonalisation: FPT $\subset$ XP.

Kernelisation I

- Before doing a ‘slow’ algorithm, first preprocess the input: build an equivalent, but smaller input.
- Kernelisation: with proof that the resulting equivalent input is *small*: size bounded by function of parameter.



$$Q(I, k) = Q(I', k')$$

- In a kernel, the size of (I', k') is bounded by a function of k.

Kernelisation II

Kernelisation algorithm

A kernel (or kernelisation algorithm) for a problem Q is an algorithm A maps inputs (I, k) of Q to inputs (I', k') of the same problem Q such that:

1. A uses polynomial time;
2. $k' \leq g(k)$ and $|I'| \leq g(k)$ for some function g (the new input has size bounded by a function of the parameter);
3. $Q(I, k) \Leftrightarrow Q(I', k')$ (the answer to the problem does not change).

Kernels and FPT

Theorem

A decidable problem P has a kernel, if and only if it fixed parameter tractable.

Proof.

$\Rightarrow$: take input (I, k), make in $|I|^{O(1)}$ time a kernel of size $f(k)$, and apply any algorithm to solve the kernel: $|I|^{O(1)} + g(f(k))$ time.
$\Leftarrow$: If P has an $f(k)n^c$ algorithm A: run A for n^{c+1} time steps. If A finishes, then output the answer (or transform to trivial instance); otherwise: $n^{c+1} < f(k)n^c \Rightarrow n < f(k)$, and the original input is a kernel. $\square$

Example: Point-line Cover

Point-line Cover

Given: n points in 2-dimensional space, integer k.
Question: Can we draw k straight lines that cover all n points?

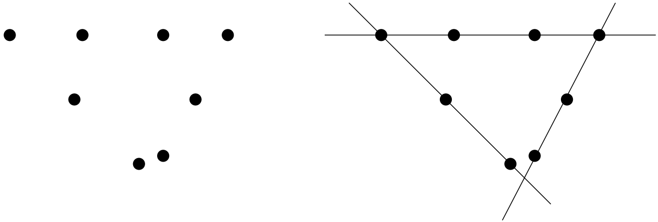


Figure: An instance, and a solution with $k = 3$

Kernelisation algorithm

1. While there is a line that covers more than k uncovered points: choose it; $k \leftarrow k - 1$.

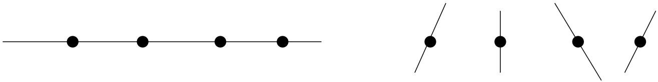


Figure: If a line covers more than k uncovered points, we must choose it

Kernelisation algorithm

1. While there is a line that covers more than k uncovered points: choose it; $k \leftarrow k - 1$.
2. If we have more than k^2 points: output NO (or give trivial input with no solution)

Observe:

- After Step 1, each line covers at most k points, so if $n > k^2$, there is no solution

Theorem

The POINT-LINE COVER *problem has a kernel with k^2 points, and is fixed parameter tractable.*

Problems without Polynomial Kernels

Kernel of polynomial size: procedure that changes (I, k) into equivalent instance of size $O(k^{O(1)})$

Theorem (BDFH+FS/D)

If a parameterized problem is compositional and with parameter in unary NP-hard, then it has no kernel of polynomial size, unless coNP $\subseteq$ NP/poly.

- Compositional comes in two flavours: or-composition (Fortnow/Santhanam) and and-composition (Drucker).
- Idea: procedure to take n instances $(I_1, k) \ldots, (I_n, k)$ and build a new instance $(I, f(k))$ with $Q(I, f(k)) \Leftrightarrow \bigvee_j Q(I_j, k)$ or $Q(I, f(k)) \Leftrightarrow \bigwedge_j Q(I_j, k)$.

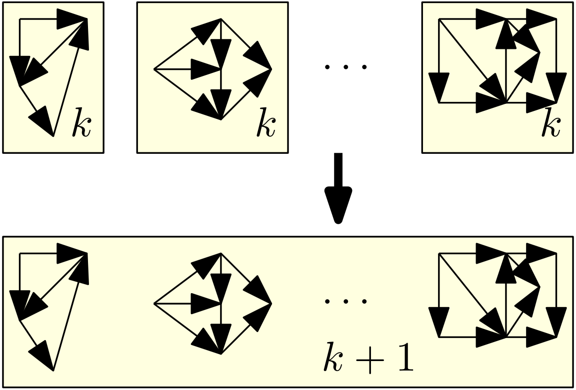
Upwards Planarity(treedepth) has no polynomial kernel

Theorem

UPWARDS PLANARITY*(treedepth) has no polynomial kernel unless* $coNP \subseteq NP/poly$.

- UPWARDS PLANARITY is NP-complete; the parameter treedepth can be given in unary.
- And-composition: take the disjoint union of graphs. The result has an upwards planar drawing, iff each component has an upwards planar drawing, and the treedepth increases by at most 1.

And-composition for Upwards Planarity(treedepth) — Illustration



XP

XP

XP (or slice-wise polynomial time) is the class of parameterized problems with an algorithm that uses $n^{f(k)}$ time for a computable function f

- Many problems known to be in XP — techniques include dynamic programming, enumerating all solutions, branching, ...
- There are a few XP-complete problems ('games with few pieces')

Distinguishing FPT and XP

- How can we tell that a problem is not in FPT?
- Using complexity classes and reductions.
- Compare to the situation P versus NP — polynomial versus exponential time.
- Downey and Fellows (1990s) introduced:
  - Parameterized reductions.
  - W-hierarchy with complexity classes: $W[1], W[2], \ldots, W[SAT], W[P]$.
    - Originally defined with help of circuits
    - Equivalent definitions with logic (follows)
  - If $W[1] = FPT$, then the Exponential Time Hypothesis is false — so, we expect that problems that are $W[1]$-hard are not Fixed Parameter Tractable.

Parameterized reduction

Complete problems are defined in terms of a type of *reductions*.

- A parameterized reduction is a function Φ that maps inputs of parameterized problem A to parameterized problem B:
  - $A(I,k) \Leftrightarrow B(\Phi(I,k))$; (YES $\Longleftrightarrow$ YES)
  - If $\Phi(I,k) = (I',k')$, then $k' \leq g(k)$ for a computable g (New parameter is also bounded);
  - $\Phi(I,k)$ can be computed in $f(k)n^{O(1)}$ time.
- Some classes have more restrictions on reductions.

Theorem (Downey,Fellows)

If A has a parameterized reduction to B, and B is in FPT, then A is in FPT.

So, if B is not in FPT, then A is not in FPT...

W[1]

- A problem belongs to $W[1]$, if and only if it has a parameterized reduction to WEIGHTED 3-SATISFIABILITY.

WEIGHTED 3-SATISFIABILITY

Given: Boolean formula F in Conjunctive Normal Form with three literals per clause, integer k.
Parameter: k.
Question: Can we satisfy F by setting exactly k variables to true and all others to false?

- Also holds also if we replace 3 by any other fixed integer ≥ 2, or 'exactly' by 'at most' or 'at least'.
- INDEPENDENT SET and CLIQUE are $W[1]$-complete; many other known $W[1]$-hard and $W[1]$-complete problems.

Upward and Orthogonal Planarity are $W[1]$-Hard Parameterized by Treewidth

Orthogonal Planarity Testing

Given: Planar graph G
Question: Is there a planar drawing where all edges are either a horizontal or a vertical segment?

Theorem (Jansen et al., GD 2023)

The UPWARD PLANARITY TESTING *and* ORTHOGONAL PLANARITY TESTING *are $W[1]$-hard with treewidth of G as parameter.*

W[2]

- $W[2]$ has similar characterisation, but clauses can be arbitrary large.
- A problem belongs to $W[2]$, if and only if it has a parameterized reduction to WEIGHTED CNF-SATISFIABILITY.

WEIGHTED CNF-SATISFIABILITY

Given: Boolean formula F in Conjunctive Normal Form, integer k
Parameter: k
Question: Can we satisfy F by setting exactly k variables to true and all others to false?

- DOMINATING SET is $W[2]$-complete.

$W[t]$ for $t > 2$

- $W[t]$ is 'roughly' problems of same complexity as deciding if a Boolean formula with t alternations between AND and OR can be satisfied by setting k variables to true.

Weighted t-Normalised Satisfiability

Given: Boolean formula F, integer k, with F of the following form (with t alternations) $\bigwedge \bigvee \bigwedge \bigvee \bigwedge \cdots (\neg) X_i$, integer k
Parameter: k
Question: Can we satisfy F by setting exactly k variables to true and all others to false?

$W[SAT]$

- $W[SAT]$: any Boolean formula.
- $W[SAT] \leftrightarrow$ WEIGHTED SATISFIABILITY.

WEIGHTED SATISFIABILITY

Given: Boolean formula F, integer k
Parameter: k
Question: Can we satisfy F by setting exactly k variables to true and all others to false?

$W[P]$

The last class in the W-hierarchy is $W[P] \leftrightarrow$ WEIGHTED CIRCUIT SATISFIABILITY.

WEIGHTED SATISFIABILITY

Given: Boolean circuit C with n input gates and one output gate, integer k
Parameter: k
Question: Can we let C output true by setting exactly k inputs to true and all others to false?

The W-hierarchy: discussion

- Hardness for $W[1]$ implies that it is unlikely that problem is FPT .
- Hardness for classes higher in W-hierarchy implies the same ('more unlikely').
- Proving W-hardness: similar to NP-completeness proofs but:
  - parameter must stay bounded;
  - exponential (or more) time in parameter is allowed.
- In W-hierarchy: problems of the form: choose (at least, at most, exactly) k elements out of n such that 'something holds'.

Completeness - for what class?

- In 'classic' complexity theory, we have many complete problems: NP-complete problems Q usually have a simple proof that $Q \in \mathbf{NP}$ and a reduction that shows that Q is **NP**-hard.
- For $W[1]$ (and other classes in the W-hierarchy), we have several problems known to be hard for the class, but are not known to be complete.
- This gives a question for many problems, known to be $W[1]$-hard: for which class are they complete? (And, does that give more insight?)

My XNLP-story starts with Bandwidth

- Well studied problem, application for Gaussian elimination. 'Reorganise a matrix such that all non-zero's are in a narrow band around the main diagonal'.

BANDWIDTH

Given: Undirected graph $G = (V, E)$, integer k
Parameter: k
Question: Is there a bijection $f : V \to \{1, 2, \ldots, |V|\}$, such that for each edge $\{v, w\} \in E$: $|(f(v) - f(w)| \leq k$?

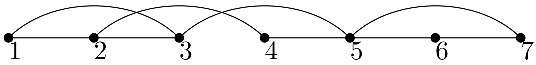


Figure: A layout with bandwidth 2

Some early results on Bandwidth

- BANDWIDTH is NP-complete, for long-hair-caterpillars with hair length three. (Monien, 1983)
- 1984, Gurari, Sudborough: BANDWIDTH is in XP: $O(n^{k+1})$ time. (Gurari, Sudborough, 1984)
- Claim by (B, Fellows, Hallett, 1994) that BANDWIDTH is $W[t]$-hard for all t for trees.
- Conjecture by Hallett, 1994: BANDWIDTH is not in $W[P]$. Main idea:
  - Problems in $W[P]$ have a certificate with $O(k \log n)$ bits.
  - Bandwidth seems to need $\Omega(n)$ bits for certificate.

More recent results on Bandwidth (parameterized)

- (Dregi, Lokshtanov, 2014): $W[1]$-hard for trees, ETH-based lower bound.
- (B, 2020): BANDWIDTH is $W[t]$-hard for all t for long-hair-caterpillars.
- (B, Groenland, Nederlof, Swennenhuis, 2021): BANDWIDTH (for long-hair-caterpillars) is XNLP-complete.

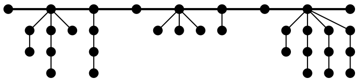


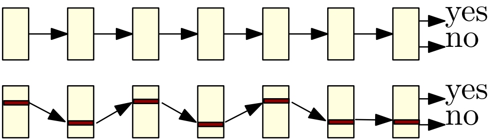
Figure: A (long-hair-)caterpillar is a tree with all vertices of degree more than two on one path

The Gurari-Sudborough (1984) Dynamic Programming algorithm for Bandwidth

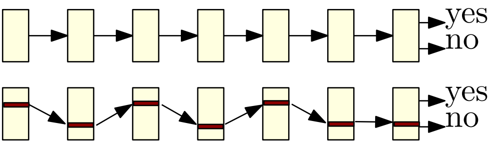
- Idea: look at initial sequences of ℓ vertices $v_1, \ldots, v_\ell$. Give such a sequence a characteristic:
  - the last k vertices, $v_{\ell-k+1}, \ldots, v_\ell$
  - which neighbours of $v_{\ell-k+1}, \ldots, v_\ell$ belong to $v_1, \ldots, v_\ell$
- If $v_1, \ldots, v_\ell$ has the same characteristic as $v'_1, \ldots, v'_\ell$ then either both or none of these two can be extended to a solution.
- For each ℓ, we have $O(n^k)$ characteristics
- Algorithm: for $\ell = 1$ to n, build table T_ℓ of all possible characteristics of initial sequences of length ℓ.
  If T_n is non-empty, say Yes, otherwise No

From an XP Algorithm to a Non-deterministic Algorithm

- Change XP-algorithm for BANDWIDTH by Gurari, Subborough to a non-deterministic algorithm:
  - Instead of building entire tables, each time non-deterministically guess one entry from each table. (In each step, guess the next vertex in the sequence, and verify if this does not conflict with the bandwidth)



A non-deterministic algorithm for Bandwidth



Algorithm has k vertices and one counter in $[1, n]$ in memory:

Lemma

Bandwidth *can be solved by a non-deterministic Turing Machine in $O(kn)$ time with $O(k \log n)$ bits additional memory.*

This brings us to a class defined by (Elberfeld et al., 2015).

Birth of XNLP

(Elberfeld, Stockhusen, Tantau, 2015) define parameterized classes with bounded memory and time, including

- $N[f\ poly, \log]$: problems solvable on
  - Non-deterministic Turing Machine;
  - $f(k)n^{O(1)}$ time;
  - $f(k) \log n$ space.
- (EST, 2015): problems complete for $N[f\ poly, \log]$:
  - Non-deterministic Turing machine acceptance with $O(k)$ cells read-write-tape (with polynomial size alphabet) and running time bounded by polynomial in n
  - Timed Non-deterministic Accepting Linear Cellular Automaton
  - Longest Common Subsequence (with variants)

(B, Groenland, Nederlof, Swennenhuis, 2021): renamed $N[f\ poly, \log]$ to XNLP.

Classes with logarithmic space

Classic

- L: deterministic, $O(\log n)$ space
- NL: non-deterministic, $O(\log n)$ space
- L and NL imply polynomial time

Parameterized

- XL: deterministic, $O(f(k) \log n)$ space
- XNL: non-deterministic, $O(f(k) \log n)$ space
- **XNLP: non-deterministic,** $O(f(k) \log n)$ **space and** $O(f(k) poly(n))$ **time**
- $\ldots \subseteq XP$

Many new XNLP-complete problems

Many new XNLP-complete problems have been found (2021 – now), with results building upon each other, including:

- Many problems with pathwidth as parameter (several papers), or with some other ‘linear width parameter’
- Reconfiguration problems (BGNS, 2021)
- Scheduling problems (BGNS 2021); (Mallem 2024)
- Linear graph structure problems, e.g., BANDWIDTH

Theorem (Blazej et al., 2024)

ORDERED LEVEL PLANARITY *Parameterized by the Number of Levels is XNLP-complete.*

Theorem (B, Groenland, Nederlof, Swennenhuis, 2021)

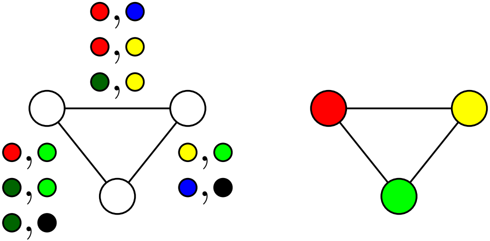
BANDWIDTH *is XNLP-complete.*

Binary CSP

Binary CSP

Given: Graph $G = (V, E)$, for each vertex v a set of colours $C(v)$, and for each edge (v, w), a set of pairs of allowed colours $C(v, w) \subseteq C(v) \times C(v)$

Question: Can we assign each vertex v a colour $f(v) \in C(v)$, such that for each edge (v, w), we have $(f(v), f(w) \in C(v, w)$?



Possible ‘starting’ XNLP-hard problem

Theorem

Binary CSP *is XNLP-complete on* $k \times n$ *grid graphs, with* k *as parameter.*

The hardness proof can be ‘generic’ (in the style of Cook’s proof of the NP-completeness of Satisfiability, using the Turing machine characterisation of the class.

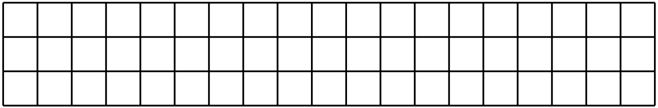


Figure: A $4 \times n$ grid graph

Proof sketch: Membership

BINARY CSP for k by n grid graphs is in XNLP:

- For $i = 1$ to n:
  - Guess the colours for the vertices in column i.
  - Have the colours of vertices in columns $i - 1$ (if existing) and i in memory.
  - Check that all adjacent vertices in columns $i - 1$ and i have allowed colour pairs. If not: reject.
- Accept.

We have $2k$ vertex colours and the value of i in memory: $O(k \log n)$ bits.

Proof sketch: Hardness I: the Turing Machine

- Finite alphabet Σ;
- Finite set of states S, with subsets S_A of accepting states and S_R of rejecting states;
- Read-Write Tape of length $f(k)\log n$ + head;
- Input tape of length n + head;
- Collection of transitions: read state, symbol at head on input tape, symbol at head at RW tape — write symbol at head on RW tape, move heads 0 or 1 step left or right, go to new state (non-deterministic).

Proof: Hardness II: Model in the grid

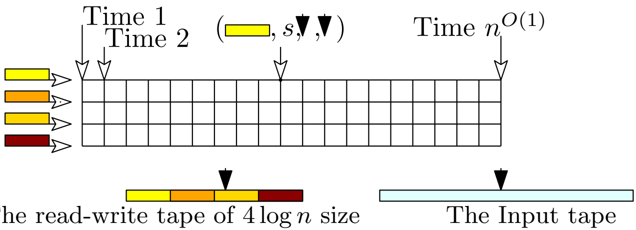
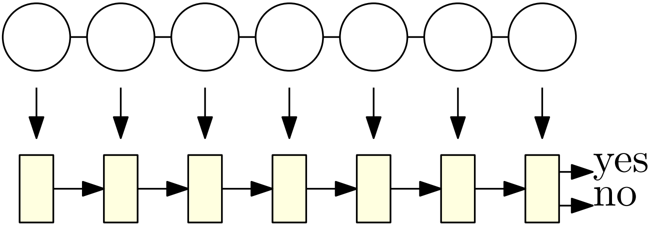


Figure: Partition the RW-tape in $f(k)$ pieces of size $\log n$ each. The colour of the vertex on row i, column t gives the content of the ith piece of RW-tape and state and location of both heads at time t.

First column colours give initial configuration; last column colours must have accepting states. BinCSP can model the proper functioning of TM.

Dynamic Programming on Path Decompositions

- Graphs of small pathwidth have a path decomposition of small width.
- Dynamic programming: compute from left to right a table for each bag.
- Deduce the answer from the last bag.



XNLP-membership Proofs on Path Decompositions

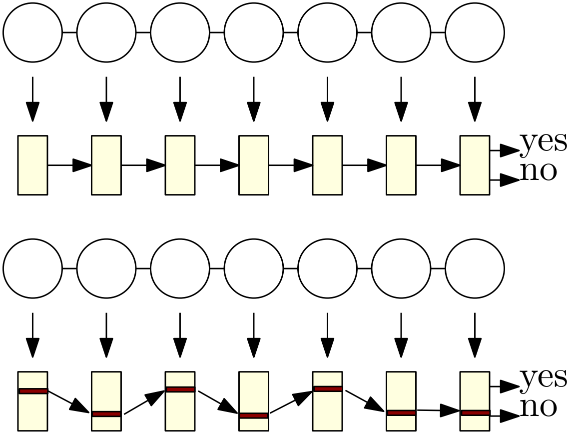


Figure: Turn the DP into XNLP-membership by guessing the element from the next table instead of building it

Example Transformation: List Colouring

List Colouring

Given: Graph $G = (V, E)$, set of colours $\mathcal{C}$, for each vertex $v \in V$, a list of colours $L(v) \subseteq \mathcal{C}$

Question: Is there a colouring $c : V \to \mathcal{C}$, such that for all $v \in V$: $c(v) \in L(v)$, and for all edges $\{v, w\} \in E$: $c(v) \neq c(w)$.

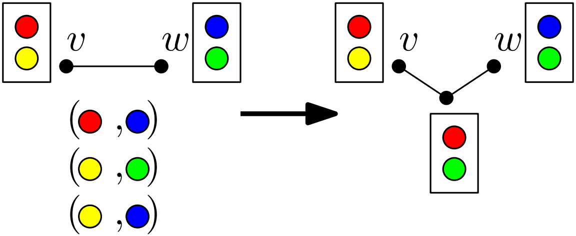
Theorem

List Colouring *with pathwidth as parameter is XNLP-complete.*

Membership with discussed technique (DP by (Jansen, Scheffler, 1997).
Hardness by reduction from Binary CSP for $k \times n$ grids.

Transformation

1. Take input of Binary CSP for $k \times n$ grids.
2. Change to equivalent instance with each vertex different colour set.
3. For each forbidden pair, add a new vertex with list the forbidden pair.



Transformation Keeps Parameter Small

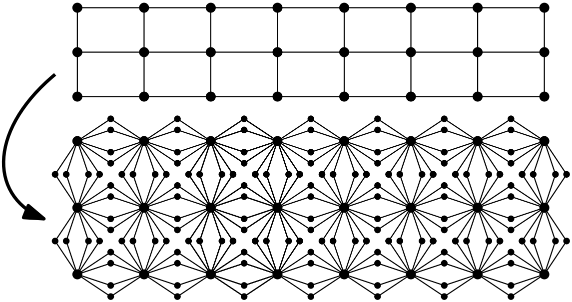


Figure: From Binary CSP to List Colouring: A $k \times n$ grid graph (pathwidth k) is transformed to a graph with pathwidth $\leq k+1$

XNLP-hardness proofs for other problems: chains of reductions, each keeping parameter bounded.

Consequences of XNLP-hardness

The Slice-wise Polynomial Space Conjecture (follows from (Michał Pilipczuk and Wrochna, 2018), building upon (Allender et al., 2014))

If Q is an XNLP-hard problem, then there is no algorithm that solves Q in $O(n^{f(k)})$ time and $f(k)n^{O(1)}$ space.

- If SPSC holds, no XP-algorithm for the problem can use FPT space!
- Indeed, the known XP algorithms for XNLP-complete problems use dynamic programming with XP-size tables.
- XNLP-hardness implies $W[t]$-hardness for all $t \in \mathbf{N}$, but with usually much simpler proofs.

From path- to tree-structured graphs

- Several problems on graphs with a linear structure are complete for XNLP.
- When parameterising by treewidth instead of pathwidth, or clique-width instead of linear clique-width, we have XNLP-hardness.
- For what class are these problems complete??

XALP

- Based on (Allender et al. 2014) and (Michał Pilipczuk and Wrochna, 2017) .
- (B, Groenland, Jacob, Pilipczuk, Pilipczuk, 2022) define a class and call it XALP (parameterized variant of class called NLPaux or SAC($O(\log n), n^c$)).
- Where XNLP characterises path-structured dynamic programming, XALP characterises tree-structured dynamic programming.

Definitions of XALP

BGJPP give a number of equivalent definitions of XALP, using Alternating Turing Machines and circuits. An intuitive definition, and easy to work with for membership proofs is:

XALP

Let XALP be the class of parameterized problems accepted by a Non-deterministic Turing Machine that

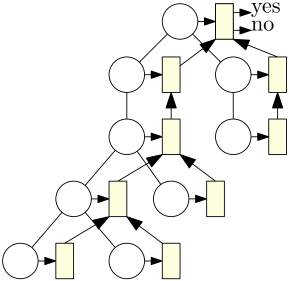
- uses $f(k)n^{O(1)}$ time, for some function f;
- has two types of memory:
  - It has a stack to which it can push symbols, or pop the top symbol;
  - It has a read-write tape of size $f(k) \log n$.

I.e., XNLP plus a stack!

Membership in XALP

- Most dynamic programming algorithms on a tree decomposition that ‘use XP time’ can be turned into XALP-membership proofs
- Change ‘build entire tables’ to ‘guess entry in each table’

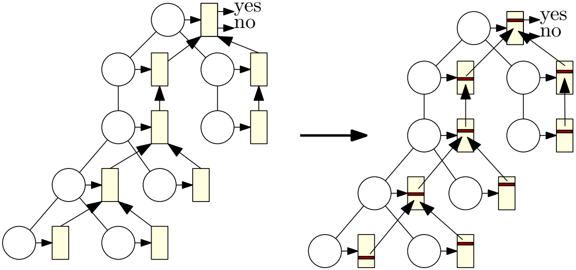
DP on tree decompositions



A dynamic programming algorithm for tree decompositions:

- computes for each bag a table, in post-order (bottom-up);
- deduces the answer from the bag of the root.

DP on tree decompositions and XALP-membership



Turn DP into XALP-membership:
Traverse tree in post-order (bottom-up).

- If bag *i* has $\alpha \in [0, 2]$ children: pop α elements from stack.
- These give ‘guessed’ table entries of children.
- From these, guess table entry for *i* and push it on stack.

XALP-complete problems

- About all problems, known to be XNLP-complete with pathwidth as parameter are XALP-complete with treewidth as parameter.
- Sequence of reductions start with
  - Non-Deterministic Turing Machine Acceptance with specific, extra conditions (details omitted here,) $\rightarrow$ Binary CSP with treewidth as parameter

Theorem (B, Szilagyi, 2024)

Binary CSP *for planar graphs with outerplanarity as parameter.*

Proof sketch: crossover gadget and embedding of graph in plane with 'few crossings'.

Proof gadget for Planar Binary CSP(outerplanarity)

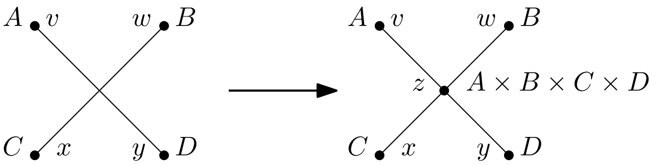


Figure: Crossover gadget.

$$
\begin{aligned}
C'(v,z) &= \{((c_1),(c_1,c_2,c_3,c_4)) \mid (c_1,c_4) \in C(v,y)\} \\
C'(w,z) &= \{((c_2),(c_1,c_2,c_3,c_4)) \mid (c_2,c_3) \in C(w,x)\} \\
C'(x,z) &= \{((c_3),(c_1,c_2,c_3,c_4)) \mid (c_2,c_3) \in C(w,x)\} \\
C'(y,z) &= \{((c_4),(c_1,c_2,c_3,c_4)) \mid (c_1,c_4) \in C(w,x)\}
\end{aligned}
$$

Construction in proof for Planar Binary CSP(outerplanarity)

- The graph in the proof for XALP-hardness for Binary CSP(treewidth) has a special structure.
- If we replace each crossover by a vertex then we get an $O(tw^2)$-outerplanar graph

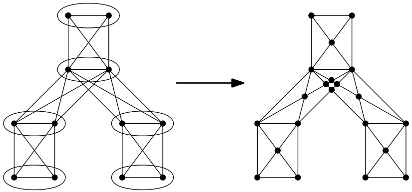


Figure: Construction gives graph with outerplanarity $O(tw^2)$

Upwards Planarit and XALP

Theorem

UPWARDS PLANARITY*(treewidth) is XALP-hard.*

Proof.

Follows from combining series of reductions:

1. BINARY CSP(treewidth) (BGJPP, 2022)
(From Turing Machine Acceptance)
2. → BINARY CSP(outerplanarity) (B, Szilagyi, 2024)
3. → ALL-OR-NOTHING FLOW(outerplanarity) (B, Szilagyi, 2024)
(elegant proof using Sidon sets (= Golomb rulers))
4. → UPWARDS PLANARITY(treewidth) (Jansen et al, GD 2023)
(Long clever proof)

□

Elements of proof: All-or-nothing flow I

All-or-Nothing Flow

Given: Directed acyclic graph $G = (V, E)$, capacity of each edge $c(e) \in \mathbf{N}$, target flow integer a, vertices s, t.
Question: Is there a flow f from s to t with value a, such that for each arc e: $f(e) = 0$ or $f(e) = c(e)$?

XALP-hardness with parameter treewidth; for planar graphs with parameter outerplanarity (or treewidth):

- From Binary CSP
- Use of Sidon set to model colours
- Two gadgets
- Piecing the gadgets together (not today)

Sidon sets / Golomb rulers

A *Sidon set* (also known as Golomb ruler) is a set of positive integers $\{a_1, \ldots, a_n\}$ with the property that each different pair of integers from the set has a different sum: $a_{i_1} + a_{i_2} = a_{j_1} + a_{j_2}$ implies $\{i_1, i_2\} = \{j_1, j_2\}$.

Theorem (Erdös, Turan, 1941)

A Golomb ruler with n elements in $\{1, 4n^2\}$ exists and can be constructed in polynomial time.

Elements of proof: All-or-nothing flow II

Map each colour to an element of a Sidon set.

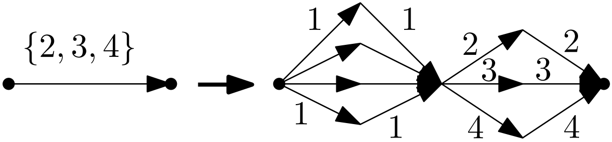


Figure: Gadget for a set S. If $2 \cdot min(S) \geq \max(S)$, then flow through gadget is 0, or one element from S

Elements of proof: All-or-nothing flow III

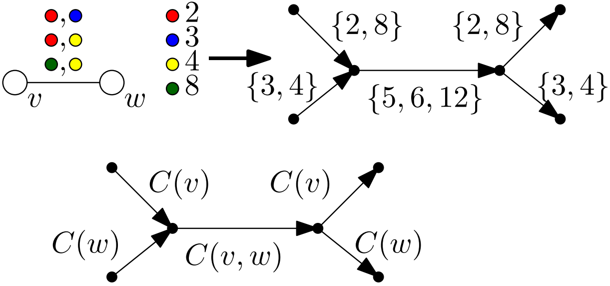


Figure: Gadget for arc *vw*. Either 0 flow, or flow models correct colour of *v*, correct colour of *w*, and correct colouring over arc *vw*. Use property of Sidon set.

More XALP-complete problems: Structural problems

- Tree Partition Width (introduced as Strong Treewidth by Seese in 1985).
- Domino Treewidth: Is there a tree decomposition of width k (parameter), such that each vertex is in at most two bags.
- Triangulating Coloured Graphs (de Vlas, 2023): Given a graph with a k-colouring of the vertices, is it a subgraph of a properly coloured chordal graph? (Problem with application from phylogeny.)

Conclusions

- Rich theory of parameterized complexity; also rich landscape of subclasses of XP.
- XNLP ‘captures’ large table sequential dynamic programming.
- XALP ‘captures’ large table tree-structured dynamic programming.
- XNLP-hardness implies (assuming the SPSC) XP-algorithms with ‘much space’.
- Many problems are known/shown to be hard for $W[1]$ or $W[2]$ — interesting to improve this to hardness for larger classes, and aim at completeness.