

# Algorithms and Networks

## Fixed-parameter tractability

Hans Bodlaender and Stefan Kratsch

March 31 & April 4, 2011



Universiteit Utrecht

1

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary

## What you have seen so far

- ▶ “easy” problems (matchings, flows, ...):
  - often nontrivial to get any polynomial runtime
  - need very intricate algorithms to achieve low runtimes
  - some sense of optimality:  $\mathcal{O}(n)$ ,  $\mathcal{O}(n \log n)$ , ...
- ▶ “hard” problems (Hamiltonian path, independent set, ...):
  - “simple” algorithms achieve runtimes like  $\mathcal{O}^*(2^n)$  or  $\mathcal{O}^*(n!)$
  - trying harder we may get  $\mathcal{O}^*(1.17^n)$  and  $\mathcal{O}^*(2^n)$  instead
  - we expect to need at least  $\mathcal{O}^*(c^n)$  but we have no clue how small  $c$  can be

Is that all? – No!



Universiteit Utrecht

2

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary

## Main idea of parameterized complexity

- ▶ measure the time complexity of algorithms not only with respect to the input size, say  $n$ , but also taking into account one or more additional parameters, say  $k$ , e.g.:

$$\mathcal{O}(2^n) \text{ vs. } \mathcal{O}(4^k n^2)$$

- ▶ no hope to be efficient for all inputs when NP-hard, but it may be possible to be efficient when  $k$  is small

Trying to restrict the “combinatorial explosion” to a parameter that is (hopefully) small on practical instances.



Universiteit Utrecht

3

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary

## Possible parameters

- ▶ solution size
- ▶ maximum degree in a graph
- ▶ treewidth of a graph
- ▶ number of literals per clause
- ▶ variety of the input: “number of numbers”
- ▶ how close is the input to a polynomial case?
- ▶ ...



Universiteit Utrecht

4

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary

## Two problems to look at

### Vertex Cover

**Input:** A graph  $G$  and an integer  $k$ .

**Output:** Does  $G$  have a vertex cover of size at most  $k$ ?

### Independent Set

**Input:** A graph  $G$  and an integer  $k$ .

**Output:** Does  $G$  have an independent set of size at least  $k$ ?

- ▶ Trivial reductions between the two problems:  
IS of size at least  $k \Leftrightarrow$  VC of size at most  $n - k$
- ▶ both can be solved in time  $\mathcal{O}(n^{k+1})$
- ▶ Vertex Cover can be solved in time  $\mathcal{O}(1.27^k + kn)$  no  $\mathcal{O}(n^{o(k)})$  known for Independent set



Introduction

Parameterized complexity

Bounded search trees

Integer linear programming

Color coding

Iterative compression

Parameterized intractability

Summary

## Polynomial for fixed $k$

- ▶ many problems (like VC and IS) can be solved in polynomial time when some relevant parameter (like solution size) is bounded by some constant
- ▶  $\mathcal{O}(n^k) \Rightarrow$  polynomial for every fixed  $k$
- ▶ still, this is quite bad for large  $k$
- ▶ can we have  $\mathcal{O}(n^c)$  for every fixed  $k$ ?  
(sometimes called uniformly polynomial for fixed  $k$ )

**Answer:** Yes, sometimes.

**Parameterized complexity:** the area that cares for such questions

**Fixed-parameter tractability:** what we just asked for



Introduction

Parameterized complexity

Bounded search trees

Integer linear programming

Color coding

Iterative compression

Parameterized intractability

Summary

## Outline

Introduction

Parameterized complexity

Bounded search trees

Integer linear programming

Color coding

Iterative compression

Parameterized intractability

Summary



Introduction

Parameterized complexity

Bounded search trees

Integer linear programming

Color coding

Iterative compression

Parameterized intractability

Summary

## Outline

Introduction

Parameterized complexity

Bounded search trees

Integer linear programming

Color coding

Iterative compression

Parameterized intractability

Summary



Introduction

Parameterized complexity

Bounded search trees

Integer linear programming

Color coding

Iterative compression

Parameterized intractability

Summary

## Classical problem

<problem name>

**Input:**

**Output:**

**formally:** classical problem  $\mathcal{L} \subseteq \Sigma^*$

(e.g. SATISFIABILITY = set (or language) of all strings that encode a satisfiable formula)

**note:** most of complexity theory is about decision problems



Universiteit Utrecht

9

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary

## Parameterized problem

<problem name>

**Input:**

**Parameter:**

**Output:**

**formally:** parameterized problem  $\mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$

**intention:** the  $\Sigma^*$  part encodes our instance as before, the integer value distinguishes some quality of our instance: e.g. solution size;

the problem should be easy for small/constant parameter values



Universiteit Utrecht

10

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary

## Examples

**Vertex Cover(k)**

**Input:** A graph  $G = (V, E)$  and an integer  $k$ .

**Parameter:**  $k$ .

**Output:** Does  $G$  have a vertex cover of size at most  $k$ ?

**Vertex Cover(tw)**

**Input:** A graph  $G = (V, E)$  and an integer  $k$ .

**Parameter:** treewidth of  $G$ .

**Output:** Does  $G$  have a vertex cover of size at most  $k$ ?

what is treewidth? – a graph parameter. details in a later lecture!



Universiteit Utrecht

11

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary

## Fixed-parameter tractability

A parameterized problem  $\mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$  is

fixed-parameter tractable if there is an algorithm that decides whether  $(x, k) \in \mathcal{Q}$  in time  $\mathcal{O}(f(k) \cdot n^c)$ , where

1.  $f$  is any computable function, and
2. the constant  $c$  is independent of  $k$ .

**FPT:** class of all fixed-parameter tractable problems

This implies a runtime of  $\mathcal{O}(n^c)$  for any constant value of  $k$ , since the  $\mathcal{O}$ -Notation “swallows” the then constant factor  $f(k)$ .

**Example:** VERTEX COVER(k) and VERTEX COVER(tw) are fixed-parameter tractable (algorithms: later!)



Universiteit Utrecht

12

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary

## Techniques to prove fixed-parameter tractability

- ▶ Bounded search trees
- ▶ Kernelization
- ▶ Iterative compression
- ▶ Integer linear programming
- ▶ Treewidth (plus other forms of dynamic programming)
- ▶ Color coding
- ▶ Algebraic techniques
- ▶ Graph Minors Theorem

**later:** parameterized intractability



Universiteit Utrecht

13

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary

## Books

- ▶ Downey & Fellows: “Parameterized Complexity” 1998
- ▶ Niedermeier: “Invitation to Fixed-Parameter Algorithms” 2006
- ▶ Flum & Grohe: “Parameterized Complexity Theory” 2006

**note:** Parameterized complexity was pioneered by Rod Downey and Michael Fellows



Universiteit Utrecht

14

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary

## Outline

Introduction

Parameterized complexity

**Bounded search trees**

Integer linear programming

Color coding

Iterative compression

Parameterized intractability

Summary



Universiteit Utrecht

15

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary

## Bounded search trees

- ▶ employ branching algorithms similar to those in the previous lecture
- ▶ analyze the size of the branching trees
- ▶ the size of the tree must be bounded by  $\mathcal{O}(f(k)n^c)$
- ▶ each branching rule must therefore decrease the parameter in all cases
- ▶ most common: parameter = solution size  
each case branched into must add something to the solution
- ▶ typically it suffices to analyze the number of recursive calls with respect to the parameter, e.g.:

$$T(k) = T(k - 1) + T(k - 3)$$



Universiteit Utrecht

16

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary

## Vertex Cover(k)

let's recall:

### Vertex Cover(k)

**Input:** A graph  $G$  and an integer  $k$ .

**Parameter:**  $k$ .

**Output:** Does  $G$  have a vertex cover of size at most  $k$ ?

- ▶ we want a set  $S \subseteq V$  of size at most  $k$
- ▶ for each  $\{u, v\} \in E$  we need  $u \in S$  or  $v \in S$
- ▶ **cannot afford** to enumerate all subsets  $S \subseteq V$ : there are  $n^k$  many



Universiteit Utrecht

17

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary

## Algorithm for Vertex Cover(k)

$VC(G = (V, E), k)$

1. if  $E = \emptyset$  return YES
2. if  $k = 0$  return NO
3. pick any edge  $\{u, v\} \in E$
4. return  $VC(G - u, k - 1)$  OR  $VC(G - v, k - 1)$

- ▶ call with  $k > 0$ , branches into 2 calls with  $k - 1$
- ▶ stop (latest) when  $k = 0$
- ▶ runtime  $\mathcal{O}(2^k \cdot n^c)$  (best known:  $\mathcal{O}(1.2738^k + kn)$ )



Universiteit Utrecht

18

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary

## Faster Algorithm for Vertex Cover(k)

$VC'(G = (V, E), k)$

1. if  $E = \emptyset$  return YES
2. if  $k = 0$  return NO
3. if maximum degree two, then compute minimum VC in polynomial time and answer accordingly
4. pick a vertex  $v$  of maximum degree ( $\deg(v) \geq 3$ )
5. return  $VC(G - v, k - 1)$  OR  $VC(G - N[v], k - \deg(v))$   
(skip the second if  $\deg(v) > k$ )

- ▶ idea: pick  $v$  or be forced to pick all neighbors of  $v$
- ▶ at least a (1,3) branching  $\Rightarrow$  branching number  $\sim 1.465$
- ▶ runtime  $\mathcal{O}(1.47^k \cdot n^c)$



Universiteit Utrecht

19

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary

## Independent Set(k)

### Independent Set(k)

**Input:** A graph  $G$  and an integer  $k$ .

**Parameter:**  $k$ .

**Output:** Does  $G$  have an independent set of size at least  $k$ ?

- ▶ again: too costly to enumerate all candidate solutions
- ▶ none of the branching rules for Vertex Cover( $k$ ) apply
- ▶ with respect to finding an independent set of size at least  $k$  the branching vectors change: there is at least one branch where we do not add anything to our solution



Universiteit Utrecht

20

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary

## Independent Set(k)

- ▶ no  $\mathcal{O}(n^{o(k)})$  algorithm is known
- ▶ in fact Independent Set(k) is W[1]-complete
- ▶ general assumption  $\text{FPT} \neq \text{W}[1]$  (like  $\text{P} \neq \text{NP}$ )
- ▶ Independent Set is FPT for some other parameters
- ▶ Independent Set(k) is FPT on special graph classes, e.g., planar graphs or graphs of bounded degree (bounded degeneracy suffices to get a simple algorithm)



## Planar Independent Set (k)

### Planar Independent Set (k)

**Input:** A planar graph  $G$  and an integer  $k$ .

**Parameter:**  $k$ .

**Output:** Does  $G$  contain an independent set of size at least  $k$ ?

**planar graph:** can be drawn in the plane with edges touching only in vertices

- ▶ NP-complete
- ▶ for any planar graph (Euler's formula):  
 $\#vertices + \#faces = \#edges + 2$
- ▶ every planar graph contains a vertex of degree at most 5  
 $\Rightarrow$  simple  $\mathcal{O}(6^k \cdot n^c)$  time algorithm



## Using the 4-color theorem

**Theorem:** Vertices of every planar graph can be colored with 4 colors such that no edge has endpoints of the same color.

- ▶ 4-color the graph  $\Rightarrow$  vertices of the same color form an independent set
- ▶ thus if  $k \leq \frac{n}{4}$  we can answer YES
- ▶ else we have  $n \leq 4k$ , hence running the fastest exact exponential time algorithm for IS costs roughly  $\mathcal{O}(1.1889^n) = \mathcal{O}(1.1889^{4k})$  (about  $\mathcal{O}(1.998^k \cdot n^c)$ )
- ▶ beating our simple degree-based algorithm by far...

**much faster:**  $\mathcal{O}(c^{\sqrt{k}} \cdot n^c)$  (maybe later)



## Cluster Editing (k)

### Cluster Editing(k)

**Input:** A graph  $G$  and an integer  $k$ .

**Parameter:**  $k$ .

**Output:** Can one add/remove at most  $k$  edges to turn  $G$  into a cluster graph?

**cluster graph:** disjoint union of cliques

- ▶ NP-complete
- ▶ cluster graph  $\Leftrightarrow$  contains no induced path on 3 vertices



## Branching algorithm for Cluster Editing ( $k$ )

$CE(G, k)$

1. if  $G$  is a cluster graph return YES
2. if  $k = 0$  return NO
3. pick a  $P_3$  in  $G$ , say  $u, v, w$ :  $u-v-w$
4. return  
 $CE(G - \{\{u, v\}\}, k - 1)$  OR  
 $CE(G - \{\{v, w\}\}, k - 1)$  OR  
 $CE(G + \{\{u, w\}\}, k - 1)$

- ▶ runtime  $\mathcal{O}(3^k \cdot n^c)$  (best known:  $\mathcal{O}(1.92^k + n^3)$ )
- ▶ how to improve? (whiteboard)



Universiteit Utrecht

25

Introduction

Parameterized complexity

Bounded search trees

Integer linear programming

Color coding

Iterative compression

Parameterized intractability

Summary

## Feedback Vertex Set( $k$ )

### Feedback Vertex Set( $k$ )

**Input:** A graph  $G = (V, E)$  and an integer  $k$ .

**Parameter:**  $k$ .

**Output:** Is there a set  $S \subseteq V$  of size at most  $k$  such that  $G - S$  is a forest?

**forest:** contains no cycles

- ▶ NP-complete
- ▶ fastest algorithm uses time  $\mathcal{O}(3.83^k n^c)$  (Cao et al. 2010)
- ▶ no obvious local branching rule (on vertices or edges)
- ▶ but:  $S$  must of course contain at least one vertex of each cycle



Universiteit Utrecht

26

Introduction

Parameterized complexity

Bounded search trees

Integer linear programming

Color coding

Iterative compression

Parameterized intractability

Summary

## Feedback Vertex Set( $k$ ) – branch on cycles

- ▶ solution(?): branch on cycles, preferably on short ones
- ▶ branching factor: length of the cycle!
- ▶ (easy) how do you find a shortest cycle in a graph?
- ▶ can we ensure that the graph has short cycles?

we will need some reduction rules



Universiteit Utrecht

27

Introduction

Parameterized complexity

Bounded search trees

Integer linear programming

Color coding

Iterative compression

Parameterized intractability

Summary

## Feedback Vertex Set( $k$ ) – reduction rules

- ▶ (easy) we may delete vertices of degree at most one
  - ▶ if  $v$  has degree two with neighbors  $u$  and  $w$  then:
    - delete  $v$  and the edges  $\{u, v\}, \{v, w\}$
    - add the edge  $\{u, w\}$
- correctness: (on whiteboard)  
key: each cycle through  $v$  also contains  $u$  and  $w$
- ▶ note that this rule may create multi-edges (convince yourself that it suffices to keep two copies of each edge)

**result:** our graph now has minimum degree three



Universiteit Utrecht

28

Introduction

Parameterized complexity

Bounded search trees

Integer linear programming

Color coding

Iterative compression

Parameterized intractability

Summary

## Feedback Vertex Set(k) – short cycles

**Theorem:** Graphs of minimum degree three and with  $n$  vertices have a shortest cycle of length at most  $2 \log n$ .

- ▶ this would give us runtime  $\mathcal{O}((2 \log n)^k n^c)$   
⇒ still too much
- ▶ Raman et al. (2006) show the following useful theorem

**Theorem:** In graphs  $G$  of minimum degree three (at least) one of the following statements hold:

1.  $G$  contains a cycle of length at most six
2.  $G$  has at most  $\mathcal{O}(k^2)$  vertices
3.  $G$  has no feedback vertex set of size at most  $k$

Introduction

Parameterized complexity

Bounded search trees

Integer linear programming

Color coding

Iterative compression

Parameterized intractability

Summary



Universiteit Utrecht

29

## Feedback Vertex Set(k) – fpt algorithm

$FVS(G, k)$

1. use our reduction rules
2. if there is a cycle of length at most six then branch on picking one of its vertices
3. if at most  $k^2$  vertices then there is a cycle of length  $4 \log k$  (first theorem on previous slide) then branch on the  $4 \log k$  choices
4. else return NO

- ▶ worst case: branching factor of  $4 \log k$
- ▶ runtime  $\mathcal{O}((4 \log k)^k n^c)$

Introduction

Parameterized complexity

Bounded search trees

Integer linear programming

Color coding

Iterative compression

Parameterized intractability

Summary



Universiteit Utrecht

30

## Outline

Introduction

Parameterized complexity

Bounded search trees

Integer linear programming

Color coding

Iterative compression

Parameterized intractability

Summary

Introduction

Parameterized complexity

Bounded search trees

Integer linear programming

Color coding

Iterative compression

Parameterized intractability

Summary



Universiteit Utrecht

31

## Linear programming

- ▶ Linear programming is a well-known tool in optimization

$$\begin{aligned} \max \quad & x_1 + 2x_2 \\ \text{s.t.} \quad & x_1 + x_2 \leq 3 \\ & 2x_1 - x_2 \leq 5 \\ & x_1, x_2 \in \mathbb{R} \end{aligned}$$

- ▶ feasibility and finding an optimal solution can be done in polynomial time

Introduction

Parameterized complexity

Bounded search trees

Integer linear programming

Color coding

Iterative compression

Parameterized intractability

Summary



Universiteit Utrecht

32

## Integer linear programming

- ▶ same as linear programming but with integer variables

$$\begin{aligned} \max & x_1 + 2x_2 \\ \text{s.t.} & \\ & x_1 + x_2 \leq 3 \\ & 2x_1 - x_2 \leq 5 \\ & x_1, x_2 \in \mathbb{N} \end{aligned}$$

- ▶ encompasses lots of problems
- ▶ (of course) NP-hard

**Theorem:** ILP with  $p$  variables can be solved in time  $\mathcal{O}(p^{cp} n^{c'})$ .  
Lenstra 1983



## An Application of ILP “with few variables”

- ▶ numerical problems parameterized by the “number of numbers”, i.e.:  
given  $a_1, \dots, a_n$  how many different values are there?
- ▶ denote the variety (number of numbers) of a set  $A = \{a_1, \dots, a_n\}$  by  $\|A\|$
- ▶ examples: Partition, Subset Sum, 3-Partition, Numerical 3-Dimensional Matching, ...  
parameterized by  $\|A\|$



## Subset Sum( $\|A\|$ )

### Subset Sum( $\|A\|$ )

**Input:** A set  $A = \{a_1, \dots, a_n\}$  and an integer  $S$ .

**Parameter:**  $\|A\|$ .

**Output:** Is there a subset  $A' \subseteq A$  whose elements sum to  $S$ ?

- ▶ NP-complete (have seen  $\mathcal{O}^*(2^{\frac{n}{2}})$  time exact algorithm)
- ▶ let  $\ell := \|A\|$
- ▶ let  $b_1, \dots, b_\ell$  denote the distinct values in  $A$
- ▶ let  $m_i$  denote how often  $b_i$  occurs in  $A$



## Subset Sum( $\|A\|$ ) – ILP

$$\begin{aligned} \max & 0 \quad \text{s.t.} \\ & \sum_{i=1}^{\ell} x_i \cdot b_i = S \\ & 0 \leq x_i \leq m_i \\ & x_i \in \mathbb{N} \end{aligned}$$

- ▶  $\ell = \|A\|$  variables  $x_1, \dots, x_\ell$  encoding how many copies of each number we take
- ▶ can solve it in time  $\mathcal{O}(\ell^{c\ell} n^{c'})$  by Lenstra's result
- ▶ note that we don't care for the target function: a feasible solution suffices



## Outline

Introduction

Parameterized complexity

Bounded search trees

Integer linear programming

**Color coding**

Iterative compression

Parameterized intractability

Summary



Universiteit Utrecht

37

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
**Color coding**  
Iterative compression  
Parameterized intractability  
Summary

## Long Path(k)

### Long Path(k)

**Input:** A graph  $G$  and an integer  $k$ .

**Parameter:**  $k$ .

**Output:** Does  $G$  contain a path of length  $k$ ?

- ▶ NP-complete (from Hamiltonian path)
- ▶ “looks like a dynamic programming problem”
- ▶ unfortunately there are roughly  $n^k$  subproblems
- ▶ main difficulty: not using vertices twice (finding walks of length  $k$  is easy)



Universiteit Utrecht

38

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
**Color coding**  
Iterative compression  
Parameterized intractability  
Summary

## Ordered Long Path(k)

### Ordered Long Path(k)

**Input:** A graph  $G$  with  $n$  vertices labeled with values  $1, \dots, k$ .

**Parameter:**  $k$ .

**Output:** Does  $G$  contain a path of length  $k$  with increasing labels?

- ▶ has a simple polynomial time algorithm
- ▶ hint: use directed edges



Universiteit Utrecht

39

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
**Color coding**  
Iterative compression  
Parameterized intractability  
Summary

## Colorful Long Path(k)

### Colorful Long Path(k)

**Input:** A graph  $G$  with vertices colored in  $k$  colors.

**Parameter:**  $k$ .

**Output:** Does  $G$  contain a path of length  $k$  in which each color occurs exactly once?

- ▶ dynamic programming might work
- ▶ subproblem:  $P(C, v)$  “is there a path using exactly one vertex of each color in  $C$  which ends in  $v$ ?” for all  $P \subseteq \{1, \dots, k\}$  and  $v \in V$
- ▶ runtime  $\mathcal{O}(2^k n^c)$



Universiteit Utrecht

40

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
**Color coding**  
Iterative compression  
Parameterized intractability  
Summary

## Long Path(k) – idea

- ▶ can we use either Ordered Long Path(k) or Colorful Long Path(k) as a subroutine to solve Long Path(k)?
- ▶ yes, both work just fine
- ▶ first we need to color the vertices of  $G$  with  $k$  colors independently at random
- ▶ chance that a path on  $k$  vertices has one vertex of each color:

$$\frac{k!}{k^k} > \frac{\left(\frac{k}{e}\right)^k}{k^k} = e^{-k}$$

- ▶ then run Colored Long Path(k)



## Long Path(k) – algorithm

### $LPath(G, k)$

1. color all vertices indep. at random with  $k$  colors
2. try to find a colored path of length  $k$
3. return YES if a path was found
4. repeat this  $t \cdot e^k$  times
5. return NO (if none of the  $t \cdot e^k$  tries was successful)

- ▶ YES answer is always correct
- ▶ NO is correct with probability at least  $1 - \left(\frac{1}{e}\right)^t$
- ▶ runtime  $\mathcal{O}(te^k 2^k n^c)$  (including runtime for Colored Long Path)
- ▶ can you make it work with Ordered Long Path?



## Note on the runtime

- ▶ if we succeed with probability  $p$  then failing  $\frac{1}{p}$  times has a probability of

$$(1 - p)^{\frac{1}{p}} < (e^{-p})^{\frac{1}{p}} = \frac{1}{e}$$

- ▶ making  $c \frac{1}{p}$  tries gives us a chance of at least  $1 - \left(\frac{1}{e}\right)^c$  of succeeding, as we fail with probability at most  $\left(\frac{1}{e}\right)^c$
- ▶  $e \sim 2.71828$  and  $\frac{1}{e} \sim 0.368$

note:

$$\lim_{x \rightarrow \infty} \left(1 + \frac{c}{x}\right)^x = \lim_{x' \rightarrow \infty} \left(1 + \frac{1}{x'}\right)^{cx'} = e^c$$

(with  $x' = \frac{x}{c}$  and using  $\lim_{z \rightarrow \infty} \left(1 + \frac{1}{z}\right)^z = e$ )



## Outline

Introduction

Parameterized complexity

Bounded search trees

Integer linear programming

Color coding

Iterative compression

Parameterized intractability

Summary



## Odd Cycle Transversal(k)

### Odd Cycle Transversal(k)

**Input:** A graph  $G$  and an integer  $k$ .

**Parameter:**  $k$ .

**Output:** Is there a set  $S$  of at most  $k$  vertices such that  $G - S$  is bipartite?

**bipartite graph:** can be 2-colored

- ▶ NP-complete
- ▶  $S$  must contain at least one vertex of each odd cycle in  $G$
- ▶ (easy) how to find an odd cycle?
- ▶ in general, odd cycles may be long and branching is expensive
- ▶ no trick known as for Feedback Vertex Set(k) (same is not possible:  $n \times n$  grid and connect opposite corners by an edge)



Universiteit Utrecht

45

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary

## Odd Cycle Transversal(k) – Compression

can we at least solve the following?

### Odd Cycle Transversal Compression(k)

**Input:** A graph  $G$ , an integer  $k$ , and a set  $S'$  of  $k + 1$  vertices such that  $G - S'$  is bipartite.

**Parameter:**  $|S'|$ .

**Output:** Is there a set  $S$  of at most  $k$  vertices such that  $G$  is bipartite?

- ▶ how to use the set  $S'$ ?
- ▶ we don't know how many vertices of  $S'$  will be in  $S$  (if any)
- ▶ we don't know what side (bipartition) the vertices in  $S'$  belong to if they are not in  $S$



Universiteit Utrecht

46

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary

## Odd Cycle Transversal(k) – Compression II

- ▶ key: we can afford to guess what will happen to  $S'$
- ▶ we try all  $3^{k+1}$  partitions of  $S'$  into
  - $T$ : vertices that are deleted, i.e., contained in  $S$
  - $L$ : vertices in left bipartition in  $G - S$
  - $R$ : vertices in right bipartition in  $G - S$
- ▶ but what now?



Universiteit Utrecht

47

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary

## A useful property of bipartite graphs

if  $G = (V, E)$  is a bipartite graph with bipartition  $V = A \cup B$  then:

- ▶ each walk with both ends in  $A$  contains an even number of edges (same for  $B$ )
- ▶ each walk with one end in  $A$  and one in  $B$  has an odd number of edges

**walk:** path that may visit vertices and edges more than once

**idea:** we will try to use the fact that there should be no odd length walks with both ends in  $L$  (or both in  $R$ ) and no even length walks with one end in  $L$  and one in  $R$  (recall our partition  $S' = L \cup R \cup T$ )



Universiteit Utrecht

48

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary

## Odd Cycle Transversal(k) – Compression III

- ▶  $G - S'$  is bipartite; let  $A \cup B$  be a bipartition of the vertices
- ▶ for any partition of  $S'$  into  $L, R, T$ :
  - let  $A_l =$  neighbors of  $L$  in  $A$
  - let  $A_r =$  neighbors of  $R$  in  $A$ ,
  - ditto for  $B_l$  and  $B_r$

**Lemma:** If deletion of a set  $X$  makes  $G - T$  bipartite such that  $L$  in left bipartition and  $R$  in right bipartition then in  $G - (S' \cup X)$  there are no paths between  $A_l$  and  $B_j$ ;  $B_r$  and  $B_j$ ;  $B_r$  and  $A_r$ ; as well as  $A_l$  and  $A_r$ .

**key idea:** such paths could, for example, be extended to odd length walks from  $L$  to  $L$



## Odd Cycle Transversal(k) – Compression IV

**Lemma:** If  $X$  is such that there are no paths between  $A_l$  and  $B_j$ ;  $B_r$  and  $B_j$ ;  $B_r$  and  $A_r$ ; as well as  $A_l$  and  $A_r$  in  $G - (S' \cup X)$  then  $G - (T \cup X)$  is bipartite with  $L$  on left bipartition and  $R$  on right bipartition.

- ▶ this is the converse of the previous lemma
- ▶ hence to find such a solution it suffices to find a set  $X$  that intersects all paths listed in the lemma
- ▶ **note:** no paths between  $A_l \cup B_r$  and  $A_r \cup B_l$
- ▶ thus for each partition of  $S'$  into  $L, R, T$  it suffices to ask for a minimum vertex cut separating these two sets (to find a minimum size set  $X$ )
- ▶ then check whether  $|T \cup X| \leq k$



## Odd Cycle Transversal(k) – Compression V

### Odd Cycle Transversal Compression(k)

**Input:** A graph  $G$ , an integer  $k$ , and a set  $S'$  of  $k + 1$  vertices such that  $G - S'$  is bipartite.

**Parameter:**  $|S'|$ .

**Output:** Is there a set  $S$  of at most  $k$  vertices such that  $G$  is bipartite?

- ▶ we can solve this problem with one maximum flow computation for each of the  $3^{|S'|} = 3^{k+1}$  partitions of  $S'$  into  $L, R$ , and  $T$
- ▶ if there is such a set  $S$ , then for one partition  $S' = L \cup R \cup T$  there must be a set  $X$  of size at most  $k - |T|$  that separates  $A_l \cup B_r$  from  $A_r \cup B_l$

**question:** how do we get a solution of size  $k + 1$ ?



## Odd Cycle Transversal(k) – Iterative Compression

- ▶ we cannot use this algorithm to compress  $V$  down to size  $|V| - 1, |V| - 2, \dots, k$ : the runtime would be  $\mathcal{O}^*(3^{|V|})$
- ▶ instead we can solve the problem for increasingly large subgraphs of  $G = (V, E)$
- ▶ let  $V = \{v_1, v_2, \dots, v_n\}$  and let  $V_t = \{v_1, v_2, \dots, v_t\}$
- ▶  $G_t$  is the subgraph of  $G$  induced by the vertex set  $V_t$

If  $G$  has an OCT of size at most  $k$  then this is true for all graphs  $G_1, \dots, G_n$ .



## Odd Cycle Transversal(k) – Iterative Compression II

1. assume that we have a solution  $S_t$  of size  $k$  for  $G_t$ , i.e.,  $G_t - S_t$  is bipartite
2. then  $S_t \cup \{v_{t+1}\}$  is a solution of size  $k + 1$  for  $G_{t+1}$ :

$$G_{t+1} - (S_t \cup \{v_{t+1}\}) = G_t - S_t$$

using that  $G_t = G_{t+1} - v_{t+1}$

3. use our compression algorithm to find a solution  $S_{t+1}$  of size  $k$  for  $G_{t+1}$
4. if not possible then return NO
5. else, repeat (using  $G_{t+1}$  and  $S_{t+1}$ )

**starting solution?:** Clearly,  $G_k$  has a solution of size  $k$ ...  
(ditto for  $G_{k+1}$  and  $G_{k+2}$ )



## Odd Cycle Transversal(k) – Final Algorithm

- ▶ solve the problem by at most  $n$  calls to the compression variant
- ▶ solve for increasing size of subinstances (but only one instance for each size)
- ▶ each time only one vertex is added so we trivially get the solution of size  $k + 1$
- ▶ runtime  $\mathcal{O}(3^k n^c)$ , i.e., at most  $3^{k+1}n$  calls to a flow algorithm
- ▶ main work: compression problem (going through all partitions of  $S'$ )

**note:** this part of the lecture was based on a paper of Lokshtanov et al. (2009): “Simpler Parameterized Algorithm for OCT” (try it, it’s only 5 pages)



## Iterative Compression – Summary

- ▶ simple but clever technique, very powerful
- ▶ central idea: maintaining a solution of size  $k$  while increasing the instance size (till original instance)
- ▶ has given the first FPT algorithm for many open problems: Odd Cycle Transversal(k), Directed Feedback Vertex Set(k), Multicut(k)
- ▶ key: can afford to guess what will happen to all elements of the given solution of size  $k + 1$
- ▶ good technique to provide the first FPT result (less applicable than Graph Minors Theorem, but reasonable runtimes)



## Outline

Introduction

Parameterized complexity

Bounded search trees

Integer linear programming

Color coding

Iterative compression

Parameterized intractability

Summary



## Parameterized intractability in a nutshell

- ▶ there is a hierarchy of parameterized complexity classes:

$$FPT \subseteq W[1] \subseteq W[2] \subseteq \dots \subseteq W[t] \subseteq \dots \subseteq W[P] \subseteq XP$$

- ▶ belief: all inclusions are strict, in particular  $FPT \neq W[1]$
- ▶ proving  $W[1]$ -hardness: main tool to rule out FPT-algorithms
- ▶  $XP$  = class of problems that can be solved in polynomial time for every fixed value of the parameter (think  $\mathcal{O}(n^{f(k)})$ )
- ▶ sometimes we can even show NP-hardness for some fixed value of  $k$  (e.g.  $\text{Coloring}(k)$ ), this implies that our problem is not in  $XP$  unless  $P = NP$

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary

**important:** What is a reasonable notion of reduction between parameterized problems?



Universiteit Utrecht

57

## Parameterized reductions in a nutshell

$Q \subseteq \Sigma^* \times \mathbb{N}, Q' \subseteq \Sigma'^* \times \mathbb{N}$  parameterized problems

**parameterized reduction:**  $\pi : (x, k) \mapsto (x', k')$  s.t. there are computable functions  $f, h$  with:

1.  $(x, k) \in Q$  iff  $(x', k') \in Q'$
2.  $k' \leq h(k)$
3.  $\pi((x, k))$  can be computed in time  $\mathcal{O}(f(k) \cdot n^c)$

we then write  $Q \leq_{fpt} Q'$

**Motivation:** when thinking about P vs. NP your reductions have P-time (so membership in P is transferred); now, we care about FPT vs.  $W[1]$  and others...  
(check that  $Q' \in FPT, Q \leq_{fpt} Q' \Rightarrow Q \in FPT$ )

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary



Universiteit Utrecht

58

## $W[1]$ in a nutshell

### Weighted q-SAT(k)

**Input:** A formula  $\mathcal{F}$  in  $q$ -CNF and an integer  $k$ .

**Parameter:**  $k$ .

**Output:** Is there a satisfying assignment for  $\mathcal{F}$  with exactly  $k$  true variables.

- ▶  $W[1]$  contains all problems that reduce to Weighted q-SAT by a parameterized reduction
- ▶ example: Independent Set(k), Clique(k), Multicolored Clique(k) are  $W[1]$ -complete
- ▶ to show  $W[1]$ -hardness: e.g. give a parameterized reduction from a  $W[1]$ -hard problem to your problem (e.g. from Weighted q-SAT(k) or Multicolored Clique(k))

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary

**note:** the standard reduction from independent set to vertex cover is not a parameterized reduction



Universiteit Utrecht

59

## $W[2]$ in a nutshell

### Weighted CNF-SAT(k)

**Input:** A formula  $\mathcal{F}$  in CNF and an integer  $k$ .

**Parameter:**  $k$ .

**Output:** Is there a satisfying assignment for  $\mathcal{F}$  with exactly  $k$  true variables.

- ▶ (again)  $W[2]$  contains all problems that reduce to Weighted CNF-SAT(k) by a parameterized reduction
- ▶ example: Dominating Set(k), Hitting Set(k), Set Cover(k)

### Dominating Set(k)

**Input:** A graph  $G$  and an integer  $k$ .

**Parameter:**  $k$ .

**Output:** Is there a set  $S$  of at most  $k$  vertices such that each other vertex has a neighbor in  $S$ ?

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary



Universiteit Utrecht

60

## Variants of Weighted q-SAT(k)

### Min Ones q-SAT(k)

**Input:** A formula  $\mathcal{F}$  in  $q$ -CNF and an integer  $k$ .

**Parameter:**  $k$ .

**Output:** Is there a satisfying assignment for  $\mathcal{F}$  with at most  $k$  true variables.

FPT by a simple  $\mathcal{O}(q^k n^c)$  time branching algorithm

### Max Ones q-SAT(k)

**Input:** A formula  $\mathcal{F}$  in  $q$ -CNF and an integer  $k$ .

**Parameter:**  $k$ .

**Output:** Is there a satisfying assignment for  $\mathcal{F}$  with at least  $k$  true variables.

NP-hard for  $k = 0$  and  $q \geq 3$  (in XP for  $q = 2$ )!

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary



Universiteit Utrecht

61

## $W[t]$ in a (high-level) nutshell

- ▶ same type of problem: satisfying a formula with an assignment that has exactly  $k$  true variables
- ▶ however: the formulas may have  $t$  alternations (2 alternations equals CNF)
- ▶ Bandwidth(k) is  $W[t]$ -hard for all  $t$  (and NP-hard on trees)

### Bandwidth(k)

**Input:** A graph  $G$  with  $n$  vertices and an integer  $k$ .

**Parameter:**  $k$ .

**Output:** Can one label the vertices of  $G$  from 1 to  $n$  such that the difference between the endpoints of any edge is at most  $k$ ?

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary



Universiteit Utrecht

62

## One last nutshell

### take away:

- ▶ reductions need to transfer tractability (e.g. reducing to an FPT problem must give you an FPT algorithm for your source problem)
- ▶ hence, parameterized reductions must ensure a restricted growth of the parameter
- ▶ we may allow them more time, i.e.,  $f(k)n^c$  (in most case  $\mathcal{O}(n^c)$  suffices)
- ▶ otherwise they behave just like the polynomial-time reductions you have used for NP-hardness

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary



Universiteit Utrecht

63

## Outline

Introduction

Parameterized complexity

Bounded search trees

Integer linear programming

Color coding

Iterative compression

Parameterized intractability

Summary

Introduction  
Parameterized complexity  
Bounded search trees  
Integer linear programming  
Color coding  
Iterative compression  
Parameterized intractability  
Summary



Universiteit Utrecht

64

## Parameterized complexity

- ▶ consider complexity with respect to input size and to some parameter value
- ▶ fixed-parameter tractability is a useful result when the parameter can be expected to be reasonably small in practice
- ▶ fairly young research area (about 20 years) but already a rich selection of techniques
- ▶ also has its own complexity theory program



## Bounded search trees

- ▶ parameterized variant of branching algorithms
- ▶ important that branching rules contribute to the solution in each case
- ▶ it can be much harder to find even a simple branching algorithm
- ▶ useful for covering problems or graph modification problems



## Integer linear programming

- ▶ applications in parameterized complexity based on a famous result of Lenstra
- ▶ he showed that ILPs with  $p$  variables can be solved in time  $\mathcal{O}(p^{cp} n^{c'})$
- ▶ useful for (integer) numerical problems, string problems, scheduling
- ▶ even a variant of Bandwidth (though with a much simpler parameter) can be solved by it



## Color coding

- ▶ randomized coloring can make it easier to find substructures
- ▶ main effect: great decrease in subproblems
- ▶ viable first approach for solving problems
- ▶ can be derandomized by using families of  $k$ -perfect families of hash functions (high level: reasonably small families of  $k$ -colorings of  $n$  element sets such that at least one member makes a given set of  $k$  elements colorful)



## Iterative compression

- ▶ key: for many problems it suffices to compress solutions of size almost  $k$  down to  $k$
- ▶ running this compression for a series of subproblems gives the final solution
- ▶ throughout this process a solution of size  $k$  is maintained after each step
- ▶ can afford to try all possibilities when compressing a solution (e.g. guess what part of it will be kept for the smaller solution)
- ▶ has settled already a number of formerly open problems in parameterized complexity



Introduction

Parameterized complexity

Bounded search trees

Integer linear programming

Color coding

Iterative compression

Parameterized intractability

Summary

# Questions?



Introduction

Parameterized complexity

Bounded search trees

Integer linear programming

Color coding

Iterative compression

Parameterized intractability

Summary