

# 3. Branching Algorithms

## COMP6741: Parameterized and Exact Computation

Serge Gaspers<sup>1,2</sup>

<sup>1</sup>School of Computer Science and Engineering, UNSW Australia

<sup>2</sup>Optimisation Research Group, NICTA

Semester 2, 2015

- 1 Introduction
- 2 Maximum Independent Set
  - Simple Analysis
  - Search Trees and Branching Numbers
  - Measure Based Analysis
  - Optimizing the measure
  - Exponential Time Subroutines
  - Structures that arise rarely
  - State Based Measures
- 3 Exercise on Max 2-CSP
- 4 Further Reading

## 1 Introduction

## 2 Maximum Independent Set

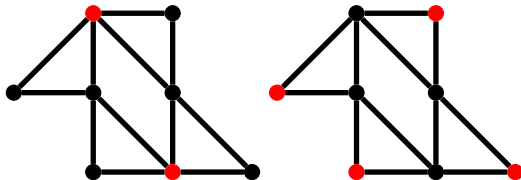
- Simple Analysis
- Search Trees and Branching Numbers
- Measure Based Analysis
- Optimizing the measure
- Exponential Time Subroutines
- Structures that arise rarely
- State Based Measures

## 3 Exercise on Max 2-CSP

## 4 Further Reading

# Recall: Maximal Independent Sets

- A vertex set  $S \subseteq V$  of a graph  $G = (V, E)$  is an **independent set** in  $G$  if there is no edge  $uv \in E$  with  $u, v \in S$ .
- An independent set is **maximal** if it is not a subset of any other independent set.
- Examples:

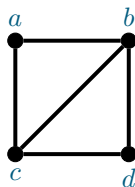


# Enumeration problem: Enumerate all maximal independent sets

ENUM-MIS

Input: graph  $G$

Output: all maximal independent sets of  $G$



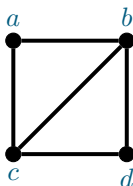
Maximal independent sets:  $\{a, d\}, \{b\}, \{c\}$

# Enumeration problem: Enumerate all maximal independent sets

ENUM-MIS

Input: graph  $G$

Output: all maximal independent sets of  $G$



Maximal independent sets:  $\{a, d\}, \{b\}, \{c\}$

**Note:** Let  $v$  be a vertex of a graph  $G$ . Every maximal independent set contains a vertex from  $N_G[v]$ .

# Branching Algorithm for ENUM-MIS

**Algorithm** `enum-mis`( $G, I$ )

**Input** : A graph  $G = (V, E)$ , an independent set  $I$  of  $G$ .

**Output**: All maximal independent sets of  $G$  that are supersets of  $I$ .

```
1  $G' \leftarrow G - N_G[I]$ 
2 if  $V(G') = \emptyset$  then                                //  $G'$  has no vertex
3   | Output  $I$ 
4 else
5   | Select  $v \in V(G')$  such that  $d_{G'}(v) = \delta(G')$  //  $v$  has min degree in  $G'$ 
6   | Run enum-mis( $G, I \cup \{u\}$ ) for each  $u \in N_{G'}[v]$ 
```

# Running Time Analysis

Define  $L(n)$  = largest number of leaves in any search tree of **enum-mis** for an instance with  $|V(G')| \leq n$ .

**Note:**  $L(n)$  is non-decreasing.



# Running Time Analysis

Define  $L(n)$  = largest number of leaves in any search tree of **enum-mis** for an instance with  $|V(G')| \leq n$ .

**Note:**  $L(n)$  is non-decreasing.

Suppose  $d_{G'}(v) = d$  generates a maximum number of leaves. Then,

$$L(n) \leq (d+1) \cdot L(n - (d+1)) = O\left((d+1)^{n/(d+1)}\right)$$

For  $s > 0$ , the function  $f(s) = s^{1/s}$  has its maximum value for  $s = e$  and for integer  $s$  the maximum value of  $f(s)$  is when  $s = 3$ .

Since the height of the search trees is  $\leq |V(G')|$ , we obtain:

## Theorem 1

Algorithm **enum-mis** has running time  $O^*(3^{n/3}) \subseteq O(1.4423^n)$ , where  $n = |V|$ .

## Corollary 2

A graph on  $n$  vertices has  $O(3^{n/3})$  maximal independent sets.

# Constraints Based Analysis

Suppose  $L(n) = 2^{\alpha \cdot n}$ ,  $\alpha > 0$ .

# Constraints Based Analysis

Suppose  $L(n) = 2^{\alpha \cdot n}$ ,  $\alpha > 0$ .

We constrain for each  $d \geq 0$ , that

$$2^{\alpha \cdot n} \geq (d + 1) \cdot 2^{\alpha \cdot (n - (d + 1))},$$

or, equivalently,

$$1 \geq (d + 1) \cdot 2^{\alpha \cdot (-(d + 1))},$$

and, since we would like to prove a small running time bound, we **minimize**  $\alpha$  subject to these constraints.

# Constraints Based Analysis

Suppose  $L(n) = 2^{\alpha \cdot n}$ ,  $\alpha > 0$ .

We constrain for each  $d \geq 0$ , that

$$2^{\alpha \cdot n} \geq (d + 1) \cdot 2^{\alpha \cdot (n - (d + 1))},$$

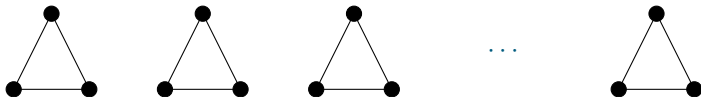
or, equivalently,

$$1 \geq (d + 1) \cdot 2^{\alpha \cdot (-(d + 1))},$$

and, since we would like to prove a small running time bound, we **minimize**  $\alpha$  subject to these constraints.

This amounts to solving a convex program, which gives  $\alpha = (1/3) \cdot \log_2 3$  and  $L(n) = 2^{(n/3) \cdot \log_2 3} = 3^{n/3}$ .

# Running Time Lower Bound



## Theorem 3

*There is an infinite family of graphs with  $\Omega(3^{n/3})$  maximal independent sets.*

## Branching Algorithm

- **Selection**: Select a local configuration of the problem instance
  - **Recursion**: Recursively solve subinstances
  - **Combination**: Compute an optimal solution of the instance based on the optimal solutions of the subinstances
- 
- **Simplification** rule: 1 recursive call
  - **Branching** rule:  $\geq 2$  recursive calls

## 1 Introduction

## 2 Maximum Independent Set

- Simple Analysis
- Search Trees and Branching Numbers
- Measure Based Analysis
- Optimizing the measure
- Exponential Time Subroutines
- Structures that arise rarely
- State Based Measures

## 3 Exercise on Max 2-CSP

## 4 Further Reading

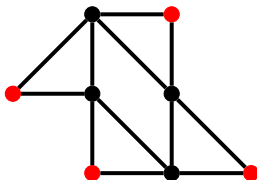


# MAXIMUM INDEPENDENT SET

## MAXIMUM INDEPENDENT SET

Input: graph  $G$

Output: A largest independent set of  $G$ .



# Exercise

Suppose there exists a  $O^*(1.2^n)$  time algorithm, which, given a graph  $G$  on  $n$  vertices, computes **the size** of a largest independent set of  $G$ .

Design an algorithm, which, given a graph  $G$ , **finds** a largest independent set of  $G$  in time  $O^*(1.2^n)$ .

# Exercise

Suppose there exists a  $O^*(1.2^n)$  time algorithm, which, given a graph  $G$  on  $n$  vertices, computes **the size** of a largest independent set of  $G$ .

Design an algorithm, which, given a graph  $G$ , **finds** a largest independent set of  $G$  in time  $O^*(1.2^n)$ .

## Solution Idea

- Compute  $k$ , the size of a largest independent set of  $G$
- Find a vertex  $v$  belonging to an independent set of size  $k$ 
  - We can do this by going through each vertex  $u$  of  $G$ , and checking whether  $G - N_G[u]$  has an independent set of size  $k - 1$
- Recurse on  $(G - N_G[v], k - 1)$

# Branching Algorithm for MAXIMUM INDEPENDENT SET

**Algorithm**  $\text{mis}(G)$

**Input** : A graph  $G = (V, E)$ .

**Output**: The size of a maximum i.s. of  $G$ .

```
1 if  $\Delta(G) \leq 2$  then                                     //  $G$  has max degree  $\leq 2$ 
2   | return the size of a maximum i.s. of  $G$  in polynomial time
3 else if  $\exists v \in V : d(v) = 1$  then                         //  $v$  has degree 1
4   | return  $1 + \text{mis}(G - N[v])$ 
5 else if  $G$  is not connected then
6   | Let  $G_1$  be a connected component of  $G$ 
7   | return  $\text{mis}(G_1) + \text{mis}(G - V(G_1))$ 
8 else
9   | Select  $v \in V$  s.t.  $d(v) = \Delta(G)$                    //  $v$  has max degree
0   | return  $\max(1 + \text{mis}(G - N[v]), \text{mis}(G - v))$ 
```

## 1 Introduction

## 2 Maximum Independent Set

- **Simple Analysis**
- Search Trees and Branching Numbers
- Measure Based Analysis
- Optimizing the measure
- Exponential Time Subroutines
- Structures that arise rarely
- State Based Measures

## 3 Exercise on Max 2-CSP

## 4 Further Reading

## Lemma 4 (Simple Analysis Lemma)

Let

- $A$  be a branching algorithm
- $\alpha > 0$ ,  $c \geq 0$  be constants

such that on input  $I$ ,  $A$  calls itself recursively on instances  $I_1, \dots, I_k$ , but, besides the recursive calls, uses time  $O(|I|^c)$ , such that

$$(\forall i : 1 \leq i \leq k) \quad |I_i| \leq |I| - 1, \text{ and} \quad (1)$$

$$2^{\alpha \cdot |I_1|} + \dots + 2^{\alpha \cdot |I_k|} \leq 2^{\alpha \cdot |I|}. \quad (2)$$

Then  $A$  solves any instance  $I$  in time  $O(|I|^{c+1}) \cdot 2^{\alpha \cdot |I|}$ .

# Simple Analysis II

## Proof.

By induction on  $|I|$ .

W.l.o.g., replace the hypotheses'  $O$  statement with a simple inequality, and for the base case assume that the algorithm returns the solution to an empty instance in time  $1 \leq |I|^{c+1}2^{\alpha \cdot |I|}$ .

Suppose the lemma holds for all instances of size at most  $|I| - 1 \geq 0$ , then the running time of algorithm  $A$  on instance  $I$  is

$$\begin{aligned}T_A(I) &\leq |I|^c + \sum_{i=1}^k T_A(I_i) && \text{(by definition)} \\ &\leq |I|^c + \sum |I_i|^{c+1} 2^{\alpha \cdot |I_i|} && \text{(by the inductive hypothesis)} \\ &\leq |I|^c + (|I| - 1)^{c+1} \sum 2^{\alpha \cdot |I_i|} && \text{(by (1))} \\ &\leq |I|^c + (|I| - 1)^{c+1} 2^{\alpha \cdot |I|} && \text{(by (2))} \\ &\leq |I|^{c+1} 2^{\alpha \cdot |I|}.\end{aligned}$$

The final inequality uses that  $\alpha \cdot |I| > 0$  and holds for any  $c \geq 0$ . □

# Simple Analysis for **mis**

- At each node of the search tree:  $O(n^2)$
- $G$  disconnected:

$$(\forall s : 1 \leq s \leq n - 1) \quad 2^{\alpha \cdot s} + 2^{\alpha \cdot (n-s)} \leq 2^{\alpha \cdot n}. \quad (3)$$

always satisfied by convexity of the function  $2^x$

- Branch on vertex of degree  $d \geq 3$

$$(\forall d : 3 \leq d \leq n - 1) \quad 2^{\alpha \cdot (n-1)} + 2^{\alpha \cdot (n-1-d)} \leq 2^{\alpha n}. \quad (4)$$

Dividing all these terms by  $2^{\alpha n}$ , the constraints become

$$2^{-\alpha} + 2^{\alpha \cdot (-1-d)} \leq 1. \quad (5)$$



## Compute optimum $\alpha$

The minimum  $\alpha$  satisfying the constraints is obtained by solving a convex mathematical program minimizing  $\alpha$  subject to the constraints (the constraint for  $d = 3$  is sufficient as all other constraints are weaker).

# Compute optimum $\alpha$

The minimum  $\alpha$  satisfying the constraints is obtained by solving a convex mathematical program minimizing  $\alpha$  subject to the constraints (the constraint for  $d = 3$  is sufficient as all other constraints are weaker).

Alternatively, set  $x := 2^\alpha$ , compute the unique positive real root of each of the **characteristic polynomials**

$$c_d(x) := x^{-1} + x^{-1-d} - 1,$$

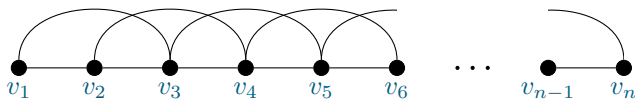
and take the maximum of these roots [Kullmann '99].

$d$	$x$	$\alpha$
3	1.3803	0.4650
4	1.3248	0.4057
5	1.2852	0.3620
6	1.2555	0.3282
7	1.2321	0.3011

# Simple Analysis: Result

- use the Simple Analysis Lemma with  $c = 2$  and  $\alpha = 0.464959$
- running time of Algorithm **mis** upper bounded by  $O(n^3) \cdot 2^{0.464959 \cdot n} = O(2^{0.4650 \cdot n})$  or  $O(1.3803^n)$

# Lower bound



$$T(n) = T(n - 5) + T(n - 3)$$

- for this graph,  $P_n^2$ , the worst case running time is  $1.1938\dots^n \cdot \text{poly}(n)$
- Run time of algo **mis** is  $\Omega(1.1938^n)$

## Mystery

What is the worst-case running time of Algorithm **mis**?

- lower bound  $\Omega(1.1938^n)$
- upper bound  $O(1.3803^n)$

## 1 Introduction

## 2 Maximum Independent Set

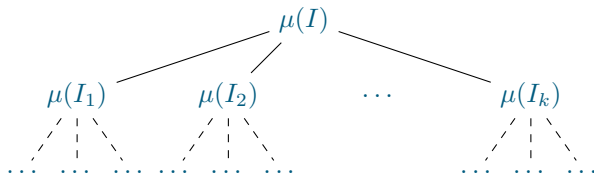
- Simple Analysis
- **Search Trees and Branching Numbers**
- Measure Based Analysis
- Optimizing the measure
- Exponential Time Subroutines
- Structures that arise rarely
- State Based Measures

## 3 Exercise on Max 2-CSP

## 4 Further Reading

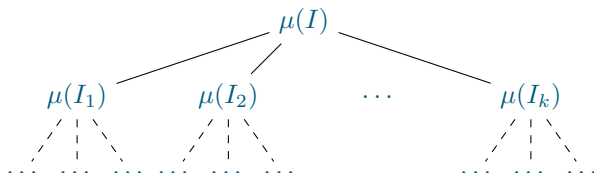
# Search Trees

Denote  $\mu(I) := \alpha \cdot |I|$ .

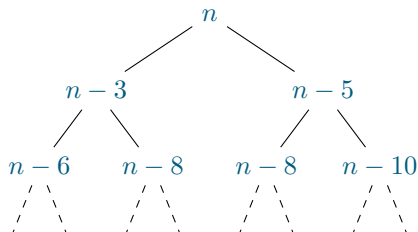


# Search Trees

Denote  $\mu(I) := \alpha \cdot |I|$ .



Example: execution of **mis** on a  $P_n^2$





# Branching number: Definition

Consider a constraint

$$2^{\mu(I)-a_1} + \dots + 2^{\mu(I)-a_k} \leq 2^{\mu(I)}.$$

Its **branching number** is

$$2^{-a_1} + \dots + 2^{-a_k},$$

and is denoted by

$$(a_1, \dots, a_k).$$

Clearly, any constraint with branching number at most 1 is satisfied.

# Branching numbers: Properties

**Dominance** For any  $a_i, b_i$  such that  $a_i \geq b_i$  for all  $i$ ,  $1 \leq i \leq k$ ,

$$(a_1, \dots, a_k) \leq (b_1, \dots, b_k),$$

as  $2^{-a_1} + \dots + 2^{-a_k} \leq 2^{-b_1} + \dots + 2^{-b_k}$ .

In particular, for any  $a, b > 0$ ,

$$\text{either } (a, a) \leq (a, b) \quad \text{or} \quad (b, b) \leq (a, b).$$

**Balance** If  $0 < a \leq b$ , then for any  $\varepsilon$  such that  $0 \leq \varepsilon \leq a$ ,

$$(a, b) \leq (a - \varepsilon, b + \varepsilon)$$

by convexity of  $2^x$ .

# Exercises

1. Let  $A$  be a branching algorithm, such that, on any input of size at most  $n$  its search tree has height at most  $n$  and for the number of leaves  $L(n)$ , we have

$$L(n) \leq 3 \cdot L(n - 2)$$

Upper bound the running time of  $A$ , assuming it spends only polynomial time at each node of the search tree.

1. Let  $A$  be a branching algorithm, such that, on any input of size at most  $n$  its search tree has height at most  $n$  and for the number of leaves  $L(n)$ , we have

$$L(n) \leq 3 \cdot L(n - 2)$$

Upper bound the running time of  $A$ , assuming it spends only polynomial time at each node of the search tree.

2. Same question, except that

$$L(n) \leq \max \begin{cases} 2 \cdot L(n - 3) \\ L(n - 2) + L(n - 4) \\ 2 \cdot L(n - 2) \\ L(n - 1) \end{cases}$$

## 1 Introduction

## 2 Maximum Independent Set

- Simple Analysis
- Search Trees and Branching Numbers
- **Measure Based Analysis**
- Optimizing the measure
- Exponential Time Subroutines
- Structures that arise rarely
- State Based Measures

## 3 Exercise on Max 2-CSP

## 4 Further Reading

- Goal, idea
  - capture more structural changes when branching into subinstances
- Means
  - potential-function method, a.k.a., **Measure & Conquer**
- Example: Algorithm **mis**
  - advantage when degrees of vertices decrease

- Model running time of **mis** by

$$T(n_1, n_2, \dots), \text{ short } T\left(\{n_i\}_{i \geq 1}\right),$$

where  $n_i := |\{v \in V : d(v) = i\}|$ .

- $G - v$ : neighbors' degrees decrease
- $G - N[v]$ : a vertex in  $N^2[v]$  has its degree decreased



## Multivariate recurrences (2)

- We obtain the following recurrence where the maximum ranges over all  $d \geq 3$ , all  $p_i, 2 \leq i \leq d$  such that  $\sum_{i=2}^d p_i = d$  and all  $k$  such that  $2 \leq k \leq d$ :

$$T\left(\{n_i\}_{i \geq 1}\right) = \max_{d, p_2, \dots, p_d, k} \begin{cases} T(\{n_i - p_i + p_{i+1} - [d = i]\}_{i \geq 1}) \\ + T(\{n_i - p_i - [d = i] - [k = i] \\ + [k = i + 1]\}_{i \geq 1}) \end{cases} \quad (6)$$

where the Iverson bracket  $[F] = \begin{cases} 1 & \text{if } F \text{ true} \\ 0 & \text{otherwise} \end{cases}$

# Solve multivariate recurrence

- restrict to max degree 5
- [Eppstein 2004]: there exists a set of weights  $w_1, \dots, w_5 \in \mathbb{R}^+$  such that a solution to (6) is within a polynomial factor of a solution to the corresponding univariate weighted model ( $T(\sum_{i=1}^5 \omega_i n_i) = \max \dots$ ).

## Definition 5

A **measure**  $\mu$  for a problem  $P$  is a function from the set of all instances for  $P$  to the set of non negative reals

$$\mu(G) := \sum_{i=1}^5 w_i n_i$$

$$(i \geq 1) \quad w_i \geq 0$$

$$(i \geq 2) \quad w_i \geq w_{i-1}$$

$$(\forall d : 2 \leq d \leq 5) \quad h_d := \min_{2 \leq i \leq d} \{w_i - w_{i-1}\}$$

By [Eppstein 2004], there exist weights  $w_i$  such that a solution to (6) corresponds to a solution to the following recurrence, where the maximum ranges over all  $d, 3 \leq d \leq 5$ , and all  $p_i, 2 \leq i \leq d$ , such that  $\sum_{i=2}^d p_i = d$ ,

$$T(\mu(G)) = \max_{d, p_2, \dots, p_d, k} \begin{cases} T\left(\mu(G) - w_d - \sum_{i=2}^d p_i \cdot (w_i - w_{i-1})\right) \\ + T\left(\mu(G) - w_d - \sum_{i=2}^d p_i \cdot w_i - h_d\right). \end{cases}$$

$$T(\mu(G)) \geq T\left(\mu(G) - w_d - \sum_{i=2}^d p_i \cdot (w_i - w_{i-1})\right) \\ + T\left(\mu(G) - w_d - \sum_{i=2}^d p_i \cdot w_i - h_d\right)$$

for all  $d, 3 \leq d \leq 5$ , and all  $p_i, 2 \leq i \leq d$ , such that  $\sum_{i=2}^d p_i = d$ .

## Lemma 6 (Measure Analysis Lemma)

Let

- $A$  be a branching algorithm
- $c \geq 0$  be a constant, and
- $\mu(\cdot), \eta(\cdot)$  be two measures for the instances of  $A$ ,

such that on input  $I$ ,  $A$  calls itself recursively on instances  $I_1, \dots, I_k$ , but, besides the recursive calls, uses time  $O(|I|^c)$ , such that

$$(\forall i) \quad \eta(I_i) \leq \eta(I) - 1, \text{ and} \quad (7)$$

$$2^{\mu(I_1)} + \dots + 2^{\mu(I_k)} \leq 2^{\mu(I)}. \quad (8)$$

Then  $A$  solves any instance  $I$  in time  $O(\eta(I)^{c+1}) \cdot 2^{\mu(I)}$ .

# Applying the lemma

$$w_i \geq 0$$

$$w_i \geq w_{i-1}$$

$$2^{\mu(G)} \geq 2^{\mu(G) - w_d - \sum_{i=2}^d p_i \cdot (w_i - w_{i-1})} + 2^{\mu(G) - w_d - \sum_{i=2}^d p_i \cdot w_i - h_d}$$

$\Leftrightarrow$

$$1 \geq 2^{-w_d - \sum_{i=2}^d p_i \cdot (w_i - w_{i-1})} + 2^{-w_d - \sum_{i=2}^d p_i \cdot w_i - h_d}$$

# Applying the lemma

$$w_i \geq 0$$

$$w_i \geq w_{i-1}$$

$$2^{\mu(G)} \geq 2^{\mu(G) - w_d - \sum_{i=2}^d p_i \cdot (w_i - w_{i-1})} + 2^{\mu(G) - w_d - \sum_{i=2}^d p_i \cdot w_i - h_d}$$

$$\Leftrightarrow$$

$$1 \geq 2^{-w_d - \sum_{i=2}^d p_i \cdot (w_i - w_{i-1})} + 2^{-w_d - \sum_{i=2}^d p_i \cdot w_i - h_d}$$

$i$	$w_i$	$h_i$
1	0	0
2	0.25	0.25
3	0.35	0.10
4	0.38	0.03
5	0.40	0.02

These values for  $w_i$  satisfy all the constraints and  $\mu(G) \leq 2n/5$  for any graph of max degree  $\leq 5$ .

Taking  $c = 2$  and  $\eta(G) = n$ , the Measure Analysis Lemma shows that **mis** has run time  $O(n^3)2^{2n/5} = O(1.3196^n)$  on graphs of max degree  $\leq 5$ .

## 1 Introduction

## 2 Maximum Independent Set

- Simple Analysis
- Search Trees and Branching Numbers
- Measure Based Analysis
- **Optimizing the measure**
- Exponential Time Subroutines
- Structures that arise rarely
- State Based Measures

## 3 Exercise on Max 2-CSP

## 4 Further Reading



# Compute optimal weights

- By convex programming [Gaspers, Sorkin 2009]

All constraints are already convex, except conditions for  $h_d$

$$(\forall d : 2 \leq d \leq 5) \quad h_d := \min_{2 \leq i \leq d} \{w_i - w_{i-1}\}$$

$\Downarrow$

$$(\forall i, d : 2 \leq i \leq d \leq 5) \quad h_d \leq w_i - w_{i-1}.$$

Use existing convex programming solvers to find optimum weights.

# convex program in AMPL

```
param maxd integer >= 3;
set DEGREES := 0..maxd;
var W {DEGREES} >= 0; # weight for vertices according to their degrees
var g {DEGREES} >= 0; # weight for degree reductions from deg i
var h {DEGREES} >= 0; # weight for degree reductions from deg \le i
var Wmax; # maximum weight of W[d]

minimize Obj: Wmax; # minimize the maximum weight

subject to MaxWeight {d in DEGREES}:
    Wmax >= W[d];
subject to gNotation {d in DEGREES : 2 <= d}:
    g[d] <= W[d]-W[d-1];
subject to hNotation {d in DEGREES, i in DEGREES : 2 <= i <= d}:
    h[d] <= W[i]-W[i-1];
subject to Deg3 {p2 in 0..3, p3 in 0..3 : p2+p3=3}:
    2^(-W[3] - p2*g[2] - p3*g[3]) + 2^(-W[3] - p2*W[2] - p3*W[3] - h[3]) <=1;
subject to Deg4 {p2 in 0..4, p3 in 0..4, p4 in 0..4 : p2+p3+p4=4}:
    2^(-W[4] - p2*g[2] - p3*g[3] - p4*g[4])
+ 2^(-W[4] - p2*W[2] - p3*W[3] - p4*W[4] - h[4]) <=1;
subject to Deg5 {p2 in 0..5, p3 in 0..5, p4 in 0..5, p5 in 0..5 :
    p2+p3+p4+p5=5}:
    2^(-W[5] - p2*g[2] - p3*g[3] - p4*g[4] - p5*g[5])
+ 2^(-W[5] - p2*W[2] - p3*W[3] - p4*W[4] - p5*W[5] - h[5]) <=1;
```

$i$	$w_i$	$h_i$
1	0	0
2	0.206018	0.206018
3	0.324109	0.118091
4	0.356007	0.031898
5	0.358044	0.002037

- use the Measure Analysis Lemma with  $\mu(G) = \sum_{i=1}^5 w_i n_i \leq 0.358044 \cdot n$ ,  $c = 2$ , and  $\eta(G) = n$
- **mis** has running time  $O(n^3)2^{0.358044 \cdot n} = O(1.2817^n)$

## 1 Introduction

## 2 Maximum Independent Set

- Simple Analysis
- Search Trees and Branching Numbers
- Measure Based Analysis
- Optimizing the measure
- **Exponential Time Subroutines**
- Structures that arise rarely
- State Based Measures

## 3 Exercise on Max 2-CSP

## 4 Further Reading

## Lemma 7 (Combine Analysis Lemma)

Let

- $A$  be a branching algorithm and  $B$  be an algorithm,
- $c \geq 0$  be a constant, and
- $\mu(\cdot), \mu'(\cdot), \eta(\cdot)$  be three measures for the instances of  $A$  and  $B$ ,

such that  $\mu'(I) \leq \mu(I)$  for all instances  $I$ , and on input  $I$ ,  $A$  either solves  $I$  by invoking  $B$  with running time  $O(\eta(I)^{c+1}) \cdot 2^{\mu'(I)}$ , or calls itself recursively on instances  $I_1, \dots, I_k$ , but, besides the recursive calls, uses time  $O(|I|^c)$ , such that

$$(\forall i) \quad \eta(I_i) \leq \eta(I) - 1, \text{ and} \quad (9)$$

$$2^{\mu(I_1)} + \dots + 2^{\mu(I_k)} \leq 2^{\mu(I)}. \quad (10)$$

Then  $A$  solves any instance  $I$  in time  $O(\eta(I)^{c+1}) \cdot 2^{\mu(I)}$ .

# Algorithm **mis** on general graphs

- use the Combine Analysis Lemma with  $A = B = \mathbf{mis}$ ,  $c = 2$ ,  
 $\mu(G) = 0.35805n$ ,  $\mu'(G) = \sum_{i=1}^5 w_i n_i$ , and  $\eta(G) = n$
- for every instance  $G$ ,  $\mu'(G) \leq \mu(G)$  because  $\forall i, w_i \leq 0.35805$
- for each  $d \geq 6$ ,

$$(0.35805, (d + 1) \cdot 0.35805) \leq 1$$

- Thus, Algorithm **mis** has running time  $O(1.2817^n)$  for graphs of arbitrary degrees

## 1 Introduction

## 2 Maximum Independent Set

- Simple Analysis
- Search Trees and Branching Numbers
- Measure Based Analysis
- Optimizing the measure
- Exponential Time Subroutines
- **Structures that arise rarely**
- State Based Measures

## 3 Exercise on Max 2-CSP

## 4 Further Reading

# Rare Configurations

- Branching on a local configuration  $C$  does not influence overall running time if  $C$  is selected only a constant number of times on the path from the root to a leaf of any search tree corresponding to the execution of the algorithm
- Can be proved formally by using measure

$$\mu'(I) := \begin{cases} \mu(I) + c & \text{if } C \text{ may be selected in the current subtree} \\ \mu(I) & \text{otherwise.} \end{cases}$$



# Avoid branching on regular instances in **mis**

```
else
  Select  $v \in V$  such that
    (1)  $v$  has maximum degree, and
    (2) among all vertices satisfying (1),  $v$  has a neighbor of
        minimum degree
  return  $\max(1 + \mathbf{mis}(G - N[v]), \mathbf{mis}(G - v))$ 
```

New measure:

$$\mu'(G) = \mu(G) + \sum_{d=3}^5 [G \text{ has a } d\text{-regular subgraph}] \cdot C_d$$

where  $C_d, 3 \leq d \leq 5$ , are constants.

# Resulting Branching numbers

For each  $d, 3 \leq d \leq 5$  and all  $p_i, 2 \leq i \leq d$  such that  $\sum_{i=2}^d p_i = d$  and  $p_d \neq d$ ,

$$\left( w_d + \sum_{i=2}^d p_i \cdot (w_i - w_{i-1}), w_d + \sum_{i=2}^d p_i \cdot w_i + h_d \right).$$

All these branching numbers are at most 1 with the optimal set of weights on the next slide

$i$	$w_i$	$h_i$
1	0	0
2	0.207137	0.207137
3	0.322203	0.115066
4	0.343587	0.021384
5	0.347974	0.004387

Thus, the modified Algorithm **mis** has running time  $O(2^{0.3480 \cdot n}) = O(1.2728^n)$ .

## 1 Introduction

## 2 Maximum Independent Set

- Simple Analysis
- Search Trees and Branching Numbers
- Measure Based Analysis
- Optimizing the measure
- Exponential Time Subroutines
- Structures that arise rarely
- State Based Measures

## 3 Exercise on Max 2-CSP

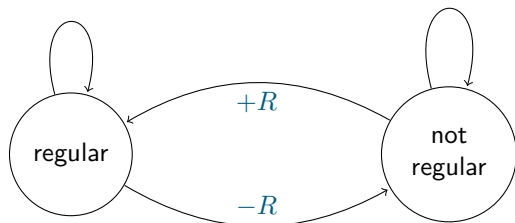
## 4 Further Reading

# State based measures

- “bad” branching always followed by “good” branchings
- **amortize** over branching numbers

$$\mu'(I) := \mu(I) + \Psi(I),$$

where  $\Psi : \mathcal{I} \rightarrow \mathbb{R}^+$  depends on global properties of the instance.



- 1 Introduction
- 2 Maximum Independent Set
  - Simple Analysis
  - Search Trees and Branching Numbers
  - Measure Based Analysis
  - Optimizing the measure
  - Exponential Time Subroutines
  - Structures that arise rarely
  - State Based Measures
- 3 Exercise on Max 2-CSP
- 4 Further Reading

# Exercise: Max 2-CSP

## MAX 2-CSP

Input: A graph  $G = (V, E)$  and a set  $S$  of *score functions* containing

- a score function  $s_e : \{0, 1\}^2 \rightarrow \mathbb{N}_0$  for each edge  $e \in E$ ,
- a score function  $s_v : \{0, 1\} \rightarrow \mathbb{N}_0$  for each vertex  $v \in V$ , and
- a score “function”  $s_\emptyset : \{0, 1\}^0 \rightarrow \mathbb{N}_0$  (which takes no arguments and is just a constant convenient for bookkeeping).

Output: The maximum score  $s(\phi)$  of an assignment  $\phi : V \rightarrow \{0, 1\}$ :

$$s(\phi) := s_\emptyset + \sum_{v \in V} s_v(\phi(v)) + \sum_{uv \in E} s_{uv}(\phi(u), \phi(v)).$$

- 1 Design simplification rules for vertices of degree  $\leq 2$ .
- 2 Using the simple analysis, design and analyze an  $O^*(2^{m/4})$  time algorithm, where  $m = |E|$ .
- 3 Use the measure  $\mu := w_e \cdot m - \left(\sum_{v \in V} w_{d_G(v)}\right)$  to improve the analysis to  $O^*(2^{m/5})$ .

# Solution sketch

## Simplification rules

S0 If there is a vertex  $y$  with  $d(y) = 0$ , then set  $s_\emptyset = s_\emptyset + \max_{C \in \{0,1\}} s_y(C)$  and delete  $y$  from  $G$ .

S1 If there is a vertex  $y$  with  $d(y) = 1$ , then denote  $N(y) = \{x\}$  and replace the instance with  $(G', S')$  where  $G' = (V', E') = G - y$  and  $S'$  is the restriction of  $S$  to  $V'$  and  $E'$  except that for all  $C \in \{0,1\}$  we set

$$s'_x(C) = s_x(C) + \max_{D \in \{0,1\}} \{s_{xy}(C, D) + s_y(D)\}.$$

S2 If there is a vertex  $y$  with  $d(y) = 2$ , then denote  $N(y) = \{x, z\}$  and replace the instance with  $(G', S')$  where  $G' = (V', E') = (V - y, (E \setminus \{xy, yz\}) \cup \{xz\})$  and  $S'$  is the restriction of  $S$  to  $V'$  and  $E'$ , except that for  $C, D \in \{0,1\}$  we set

$$s'_{xz}(C, D) = s_{xz}(C, D) + \max_{F \in \{0,1\}} \{s_{xy}(C, F) + s_{yz}(F, D) + s_y(F)\}$$

if there was already an edge  $xz$ , discarding the first term  $s_{xz}(C, D)$  if there was not.



# Solution sketch II

## Branching rules

- B Let  $y$  be a vertex of maximum degree. There is one subinstance  $(G', s^C)$  for each color  $C \in \{0, 1\}$ , where  $G' = (V', E') = G - y$  and  $s^C$  is the restriction of  $s$  to  $V'$  and  $E'$ , except that we set

$$(s^C)_\emptyset = s_\emptyset + s_y(C),$$

and, for every neighbor  $x$  of  $y$  and every  $D \in \{0, 1\}$ ,

$$(s^C)_x(D) = s_x(D) + s_{xy}(D, C).$$

- Branching on a vertex of degree  $\geq 4$  removes  $\geq 4$  edges from both subinstances
- Branching on a vertex of degree 3 removes  $\geq 6$  edges from both subinstances since  $G$  is 3-regular.

The recurrence  $T(m) \leq 2 \cdot T(m - 4)$  solves to  $2^{m/4}$

# Solution sketch III

Using the measure

$$\mu := w_e \cdot m + \left( \sum_{v \in V} w_{d_G(v)} \right)$$

we constrain that

$w_d \leq 0$	for all $d \geq 0$ to ensure that $\mu \leq w_e m$
$d \cdot w_e / 2 + w_d \geq 0$	for all $d \geq 0$ to ensure that $\mu(G) \geq 0$
$-w_0 \leq 0$	constraint for S0
$-w_2 - w_e \leq 0$	constraint for S2

$$1 - w_d - d \cdot w_e - d \cdot (w_j - w_{j-1}) \leq 0$$

for all  $d, j \geq 3$ .

Using  $w_e = 0.2$ ,  $w_0 = 0$ ,  $w_1 = -0.05$ ,  $w_2 = -0.2$ ,  $w_3 = -0.05$ , and  $w_d = 0$  for  $d \geq 4$ , all constraints are satisfied and  $\mu(G) \leq m/5$  for each graph  $G$ .

## 1 Introduction

## 2 Maximum Independent Set

- Simple Analysis
- Search Trees and Branching Numbers
- Measure Based Analysis
- Optimizing the measure
- Exponential Time Subroutines
- Structures that arise rarely
- State Based Measures

## 3 Exercise on Max 2-CSP

## 4 Further Reading

- Chapter 2, *Branching* in  
Fedor V. Fomin and Dieter Kratsch. Exact Exponential Algorithms. Springer, 2010.
- Chapter 6, *Measure & Conquer* in  
Fedor V. Fomin and Dieter Kratsch. Exact Exponential Algorithms. Springer, 2010.
- Chapter 2, *Branching Algorithms* in  
Serge Gaspers. Exponential Time Algorithms: Structures, Measures, and Bounds. VDM Verlag Dr. Mueller, 2010.