# Advice Complexity of Adaptive Priority Algorithms: Part 2 – Lower Bounds

Joan Boyar[1*], Kim S. Larsen[1], Denis Pankratov[2]

1 University of Southern Denmark
2 Concordia University

OLAWA 2020

# Overview

1. Old lower bound for priority algorithms without advice
2. Gadget pairs
3. Lower bound for optimality — Vertex Cover
4. Lower bounds for approximation
    1. Template for proving hardness results
    2. Example results

Section 1

# Old Lower Bound Result for Priority Algorithms without Advice

# Adaptive Priority Algorithms for Vertex Cover

**Theorem (Borodin,B.,Larsen,Mirmohammadi, 2010)**

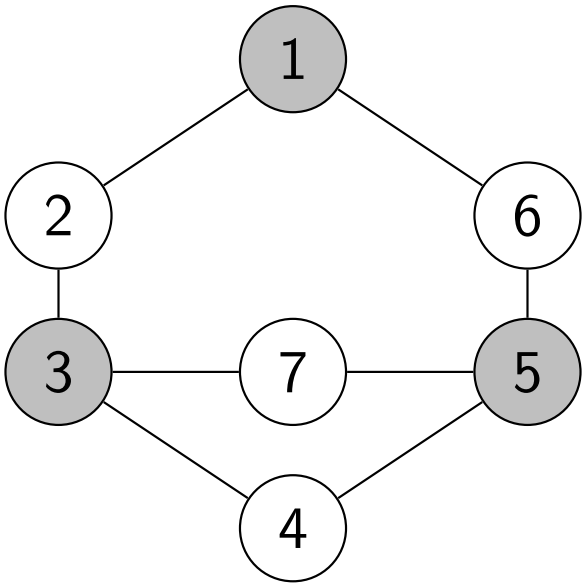
For Vertex Cover, no adaptive priority algorithm can achieve an approximation ratio better than 4/3.

# Adaptive Priority Algorithms for Vertex Cover

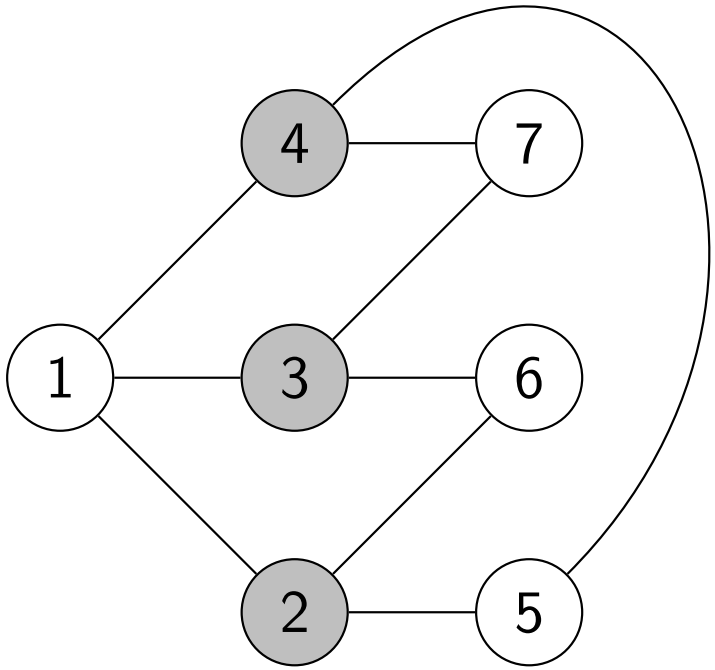
**Theorem (Borodin,B.,Larsen,Mirmohammadi, 2010)**

For Vertex Cover, no adaptive priority algorithm can achieve an approximation ratio better than 4/3.

**Pf.**




Graph $G_1$



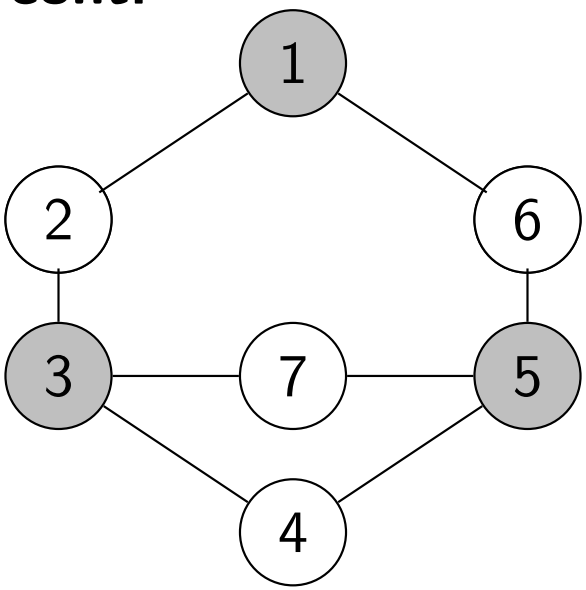

Graph $G_2$

# Adaptive Priority Algorithms for Vertex Cover

**Theorem (Borodin,B.,Larsen,Mirmohammadi, 2010)**

For Vertex Cover, no adaptive priority algorithm can achieve an approximation ratio better than 4/3.

**Pf. cont.**




Graph $G_1$

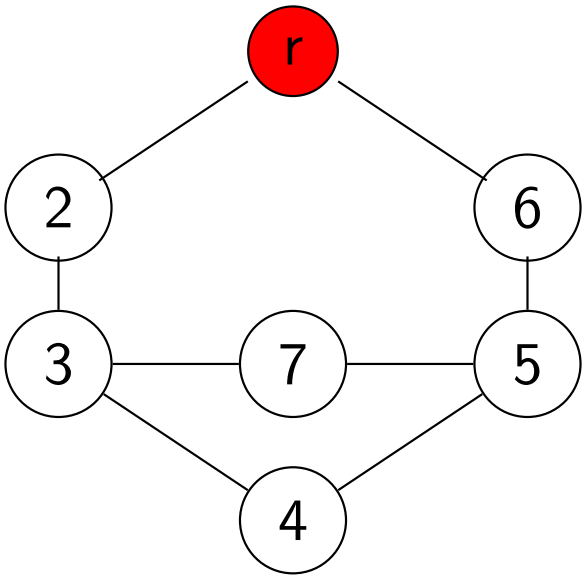
In $G_1$, rejecting vertex 1, gives a vertex cover of size 4.

# Adaptive Priority Algorithms for Vertex Cover

**Theorem (Borodin,B.,Larsen,Mirmohammadi, 2010)**

For Vertex Cover, no adaptive priority algorithm can achieve an approximation ratio better than 4/3.

**Pf. cont.**




Graph $G_1$

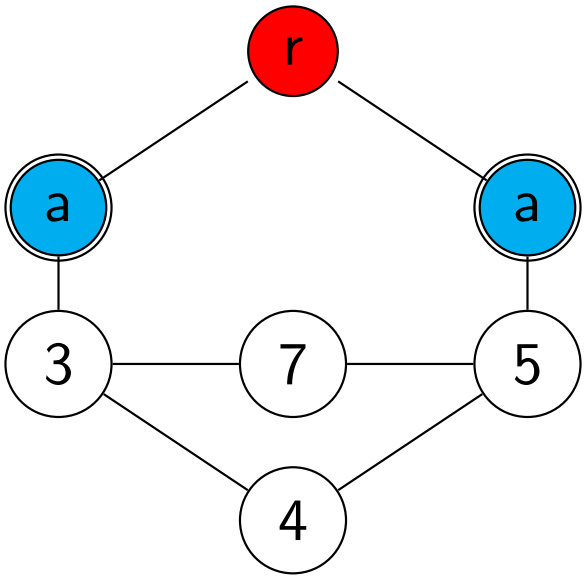
In $G_1$, rejecting vertex 1, gives a vertex cover of size 4.

# Adaptive Priority Algorithms for Vertex Cover

**Theorem (Borodin,B.,Larsen,Mirmohammadi, 2010)**

For Vertex Cover, no adaptive priority algorithm can achieve an approximation ratio better than 4/3.

**Pf. cont.**




Graph $G_1$

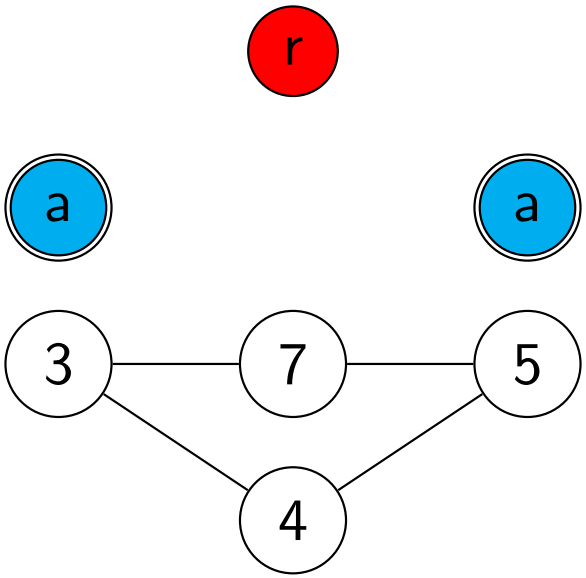
In $G_1$, rejecting vertex 1, gives a vertex cover of size 4.

# Adaptive Priority Algorithms for Vertex Cover

**Theorem (Borodin,B.,Larsen,Mirmohammadi, 2010)**

For Vertex Cover, no adaptive priority algorithm can achieve an approximation ratio better than 4/3.

**Pf. cont.**




Graph $G_1$

In $G_1$, rejecting vertex 1, gives a vertex cover of size 4.

# Adaptive Priority Algorithms for Vertex Cover

**Theorem (Borodin,B.,Larsen,Mirmohammadi, 2010)**

No adaptive priority algorithm can achieve an approximation ratio better than 4/3 for the Vertex Cover problem.

**Pf. cont.** (Adversary argument)
Input items are (Vertex name, Names of neighbors).

# Adaptive Priority Algorithms for Vertex Cover

**Theorem (Borodin,B.,Larsen,Mirmohammadi, 2010)**

No adaptive priority algorithm can achieve an approximation ratio better than 4/3 for the Vertex Cover problem.

**Pf. cont.** (Adversary argument)
Input items are (Vertex name, Names of neighbors).

Input will be an isomorphic copy of $G_1$ or $G_2$.
The input universe contains all possible input items consistent with that.

# Adaptive Priority Algorithms for Vertex Cover

**Theorem (Borodin,B.,Larsen,Mirmohammadi, 2010)**

No adaptive priority algorithm can achieve an approximation ratio better than 4/3 for the Vertex Cover problem.

**Pf. cont.** (Adversary argument)
Input items are (Vertex name, Names of neighbors).

Input will be an isomorphic copy of $G_1$ or $G_2$.
The input universe contains all possible input items consistent with that.

If ALG's first priority function selects some vertex $v$, the adversary, ADV, can make it be any vertex of the same degree, in either $G_1$ or $G_2$.

# Adaptive Priority Algorithms for Vertex Cover

**Theorem (Borodin,B.,Larsen,Mirmohammadi, 2010)**

No adaptive priority algorithm can achieve an approximation ratio better than 4/3 for the Vertex Cover problem.

**Pf. cont.** (Adversary argument)
Input items are (Vertex name, Names of neighbors).

Input will be an isomorphic copy of $G_1$ or $G_2$.
The input universe contains all possible input items consistent with that.

If ALG's first priority function selects some vertex $v$, the adversary, ADV, can make it be any vertex of the same degree, in either $G_1$ or $G_2$.

- If $v$ has degree 2 and ALG accepts, ADV chooses vertex 2 in $G_1$.
- If $v$ has degree 2 and ALG rejects, ADV chooses vertex 1 in $G_1$.

# Adaptive Priority Algorithms for Vertex Cover

**Theorem (Borodin,B.,Larsen,Mirmohammadi, 2010)**

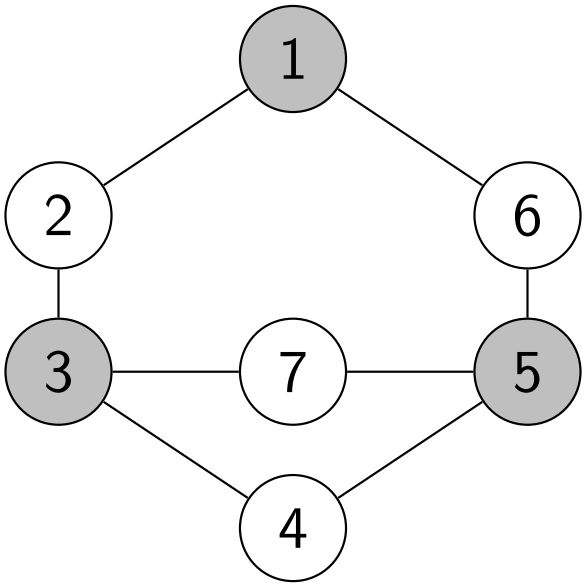
No adaptive priority algorithm can achieve an approximation ratio better than 4/3 for the Vertex Cover problem.

**Pf. cont.** (Adversary argument)
Input items are (Vertex name, Names of neighbors).

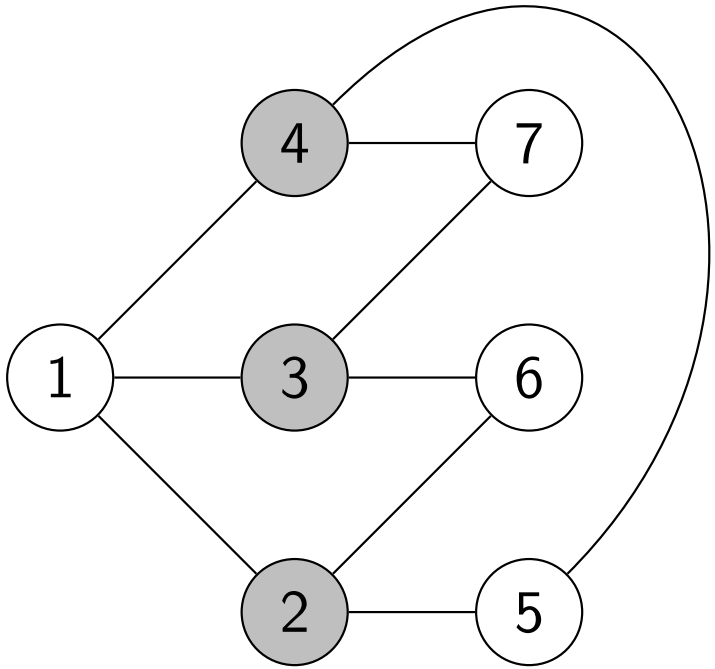
Input will be an isomorphic copy of $G_1$ or $G_2$.
The input universe contains all possible input items consistent with that.

If ALG's first priority function selects some vertex $v$, the adversary, ADV, can make it be any vertex of the same degree, in either $G_1$ or $G_2$.

- If $v$ has degree 2 and ALG accepts, ADV chooses vertex 2 in $G_1$.
- If $v$ has degree 2 and ALG rejects, ADV chooses vertex 1 in $G_1$.
- If $v$ has degree 3, and ALG accepts, ADV chooses vertex 1 in $G_2$.
- If $v$ has degree 3 and ALG rejects, ADV chooses vertex 3 in $G_1$.




Graph $G_1$




Graph $G_2$

# Adaptive Priority Algorithms for Vertex Cover

**Theorem ([Borodin,B.,Larsen,Mirmohammadi, 2010)**

For Vertex Cover, no adaptive priority algorithm can achieve an approximation ratio better than 4/3.

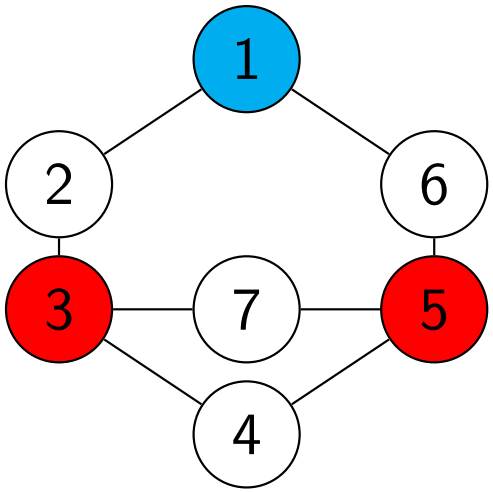
**Pf. cont.**

In all cases, ALG accepts $\geq 4$ vertices, but 3 is optimal. $\square$

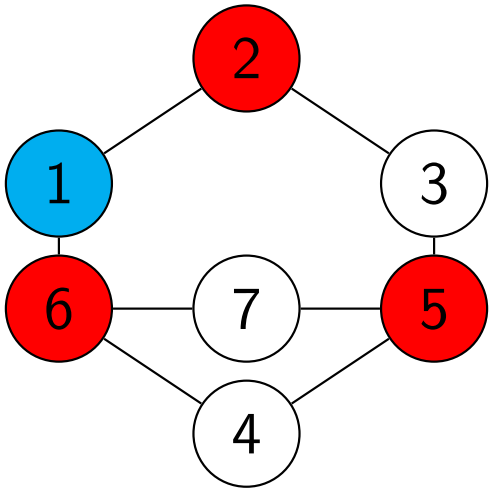
Section 2

# Gadget Pairs

# Gadget pairs



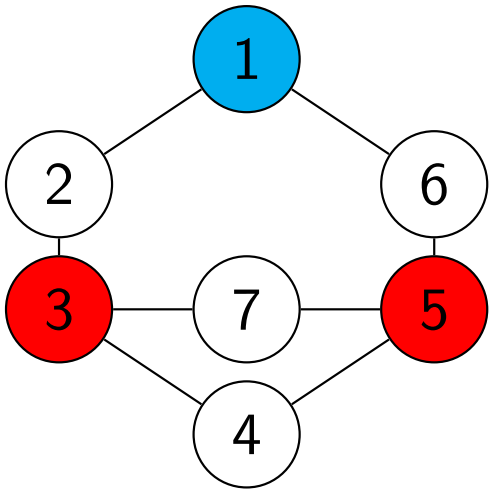

Graph $G_1$



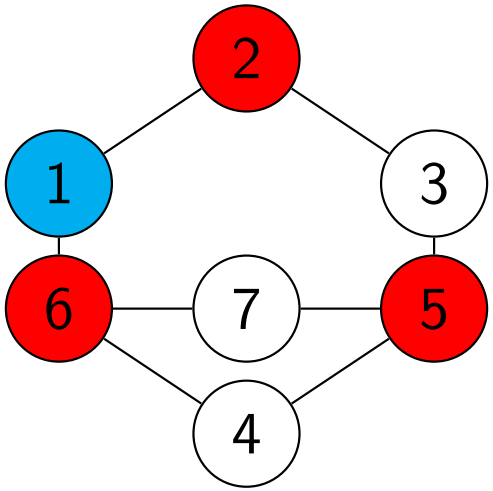

Graph $G_1'$

$G_1$ and $G_1'$ are a gadget pair
(degree-2 chosen first – call $v$ – vertex 1).

# Gadget pairs




Graph $G_1$

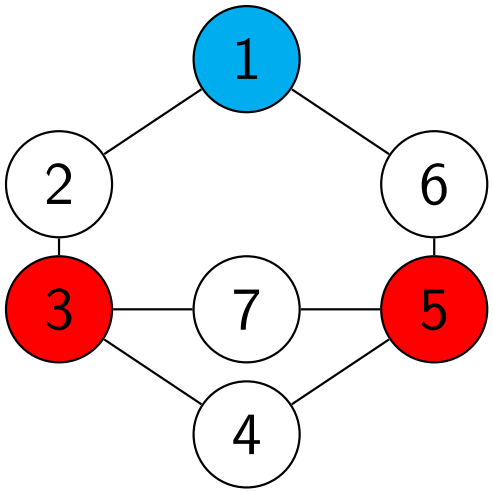



Graph $G_1'$

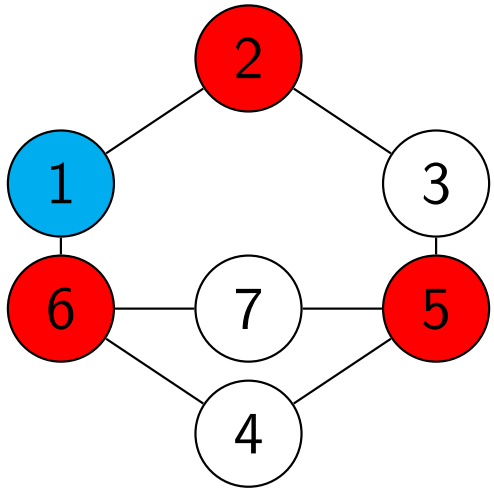
$G_1$ and $G_1'$ are a gadget pair
(degree-2 chosen first – call $v$ – vertex 1).

Properties:

# Gadget pairs




Graph $G_1$



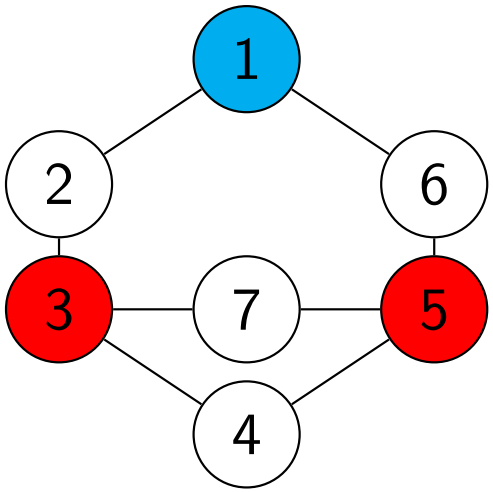

Graph $G_1'$

$G_1$ and $G_1'$ are a gadget pair
(degree-2 chosen first – call $v$ – vertex 1).

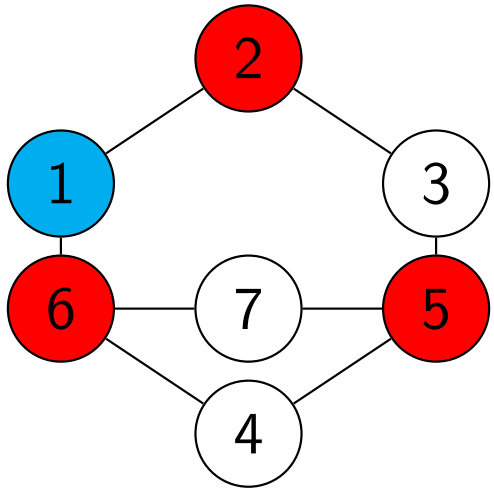
Properties:

First item condition: Input item for $v$ gives no info as to which of $G_1/G_1'$.

# Gadget pairs




Graph $G_1$




Graph $G_1'$

$G_1$ and $G_1'$ are a gadget pair
(degree-2 chosen first – call $v$ – vertex 1).

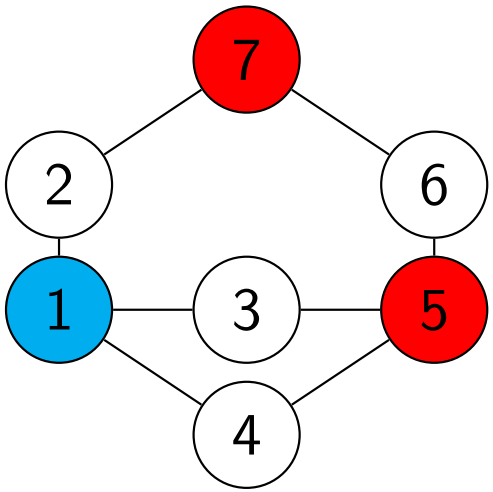
Properties:

First item condition: Input item for $v$ gives no info as to which of $G_1/G_1'$.

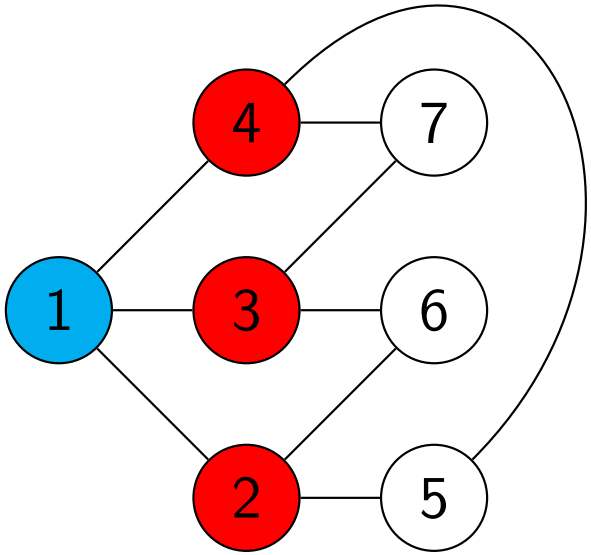
Distinguishing decision condition:

Decisions for $v$ giving optimum for $G_1$ and $G_1'$ are opposite.

# Gadget pairs



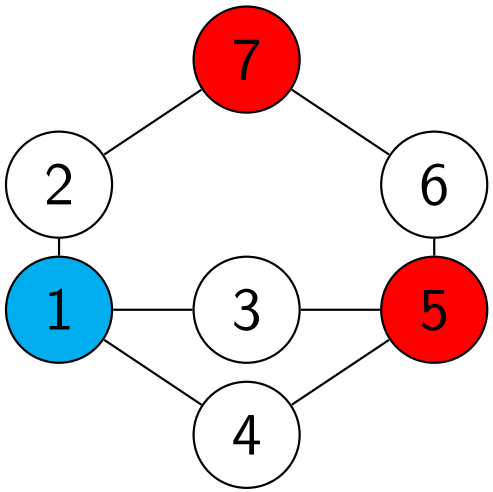

Graph $G_1$



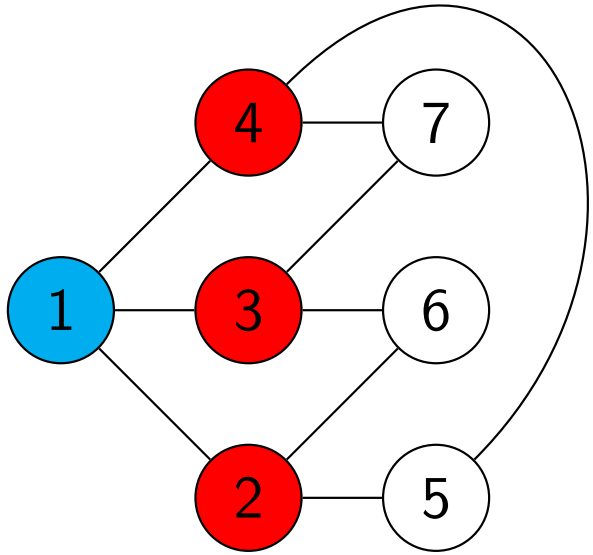

Graph $G_2$

$G_1$ and $G_2$ are a gadget pair
(degree-3 chosen first – call $v$ – vertex 1).

# Gadget pairs




Graph $G_1$

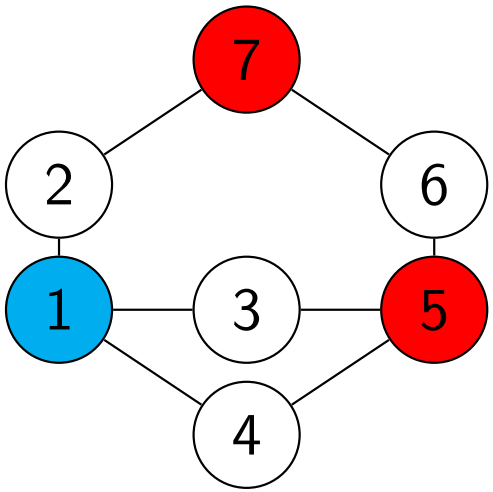



Graph $G_2$

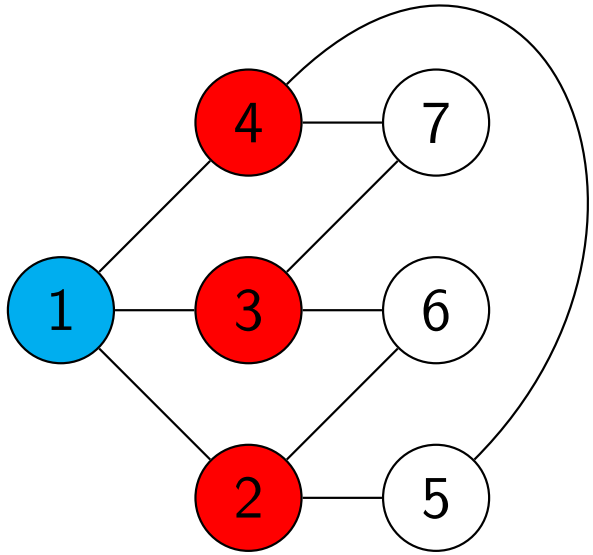
$G_1$ and $G_2$ are a gadget pair
(degree-3 chosen first – call $v$ – vertex 1).

Properties:

# Gadget pairs




Graph $G_1$



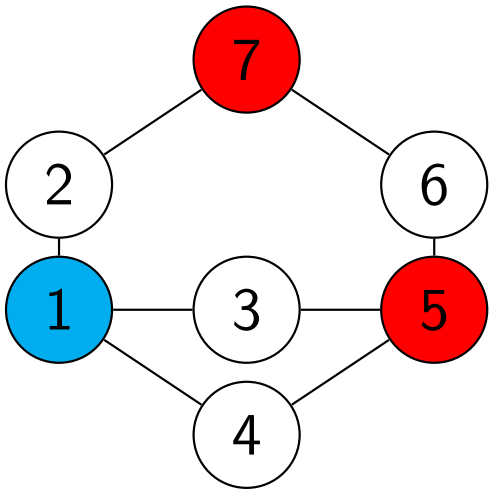

Graph $G_2$

$G_1$ and $G_2$ are a gadget pair
(degree-3 chosen first – call $v$ – vertex 1).

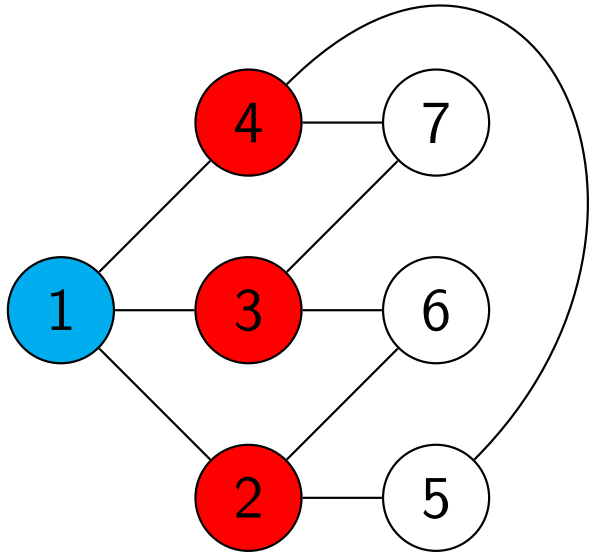
Properties:

First item condition: Input item for $v$ gives no info as to which of $G_1/G_2$.

# Gadget pairs




Graph $G_1$




Graph $G_2$

$G_1$ and $G_2$ are a gadget pair
(degree-3 chosen first – call $v$ – vertex 1).

Properties:
First item condition: Input item for $v$ gives no info as to which of $G_1/G_2$.
Distinguishing decision condition:
Decisions for $v$ giving optimum for $G_1$ and $G_2$ are opposite.

Section 3

# Lower Bound for Optimality – Vertex Cover

# Vertex Cover – lower bound for optimality

**Theorem**

For Vertex Cover, in order to achieve optimality, an adaptive priority algorithm with advice (Model 2) requires at least $\lfloor |V|/7 \rfloor$ bits of advice.

**Pf.** Create $m$ disjoint universes for $m$ gadgets, so the resulting input can be $H_1, H_2, \ldots, H_m$, where $H_i$ is an isomorphic copy of either $G_1$ or $G_2$.

# Vertex Cover – lower bound for optimality

**Theorem**

For Vertex Cover, in order to achieve optimality, an adaptive priority algorithm with advice (Model 2) requires at least $\lfloor |V|/7 \rfloor$ bits of advice.

**Pf.** Create $m$ disjoint universes for $m$ gadgets, so the resulting input can be $H_1, H_2, \ldots, H_m$, where $H_i$ is an isomorphic copy of either $G_1$ or $G_2$.

Informally, the input items for $H_i$ could require $\textsc{Alg}$ to accept the first vertex of $H_i$ or to reject it.

# Vertex Cover – lower bound for optimality

**Theorem**

For Vertex Cover, in order to achieve optimality, an adaptive priority algorithm with advice (Model 2) requires at least $\lfloor |V|/7 \rfloor$ bits of advice.

**Pf.** Create $m$ disjoint universes for $m$ gadgets, so the resulting input can be $H_1, H_2, \ldots, H_m$, where $H_i$ is an isomorphic copy of either $G_1$ or $G_2$.

Informally, the input items for $H_i$ could require $\textsc{Alg}$ to accept the first vertex of $H_i$ or to reject it.

One bit of advice is needed for the first vertex of $H_i$, $1 \leq i \leq m$.

# Vertex Cover – lower bound for optimality

**Theorem**

For Vertex Cover, in order to achieve optimality, an adaptive priority algorithm with advice (Model 2) requires at least $\lfloor |V|/7 \rfloor$ bits of advice.

**Pf.** Create $m$ disjoint universes for $m$ gadgets, so the resulting input can be $H_1, H_2, \ldots, H_m$, where $H_i$ is an isomorphic copy of either $G_1$ or $G_2$.

Informally, the input items for $H_i$ could require $\textsc{Alg}$ to accept the first vertex of $H_i$ or to reject it.

One bit of advice is needed for the first vertex of $H_i$, $1 \leq i \leq m$.

Each $H_i$ has 7 vertices, so $m = |V|/7$. $\square$

# General theorem – lower bound for optimality

**Theorem**

Let $B$ be a problem with some reasonable properties and gadget pairs. Let $s = \max_j(|G_j^a|, |G_j^r|)$, where the cardinality of a gadget is the number of input items it consists of.

Any optimal adaptive priority algorithm with advice (Model 2) for $B$ must use at least $\lfloor n/s \rfloor$ advice bits on worst case instances with $n$ input items.

Section 4

# Lower Bounds for Approximation

# Approximation – Binary String Guessing Problem

Binary string guessing problem (with known history): 2-SGKH
[Emek,Fraigniaud,Korman,Rosén, 2011]
[Böckenhauer,Hromkovič,Komm,Krug,Smula,Sprock, 2014]

# Approximation – Binary String Guessing Problem

Binary string guessing problem (with known history): 2-SGKH
[Emek,Fraigniaud,Korman,Rosén, 2011]
[Böckenhauer,Hromkovič,Komm,Krug,Smula,Sprock, 2014]

- Guess the next bit in a bit string revealed in an online manner

# Approximation – Binary String Guessing Problem

Binary string guessing problem (with known history): 2-SGKH
[Emek,Fraigniaud,Korman,Rosén, 2011]
[Böckenhauer,Hromkovič,Komm,Krug,Smula,Sprock, 2014]

- Guess the next bit in a bit string revealed in an online manner
- $\langle 0,\ 1,\ 0,\ ? \rangle$

# Approximation – Binary String Guessing Problem

Binary string guessing problem (with known history): 2-SGKH
[Emek,Fraigniaud,Korman,Rosén, 2011]
[Böckenhauer,Hromkovič,Komm,Krug,Smula,Sprock, 2014]

- Guess the next bit in a bit string revealed in an online manner
- $\langle 0,\ 1,\ 0,\ ? \rangle$
- A linear amount advice is required to make mistakes on fewer than half of the bits.

# Approximation – Binary String Guessing Problem

Binary string guessing problem (with known history): 2-SGKH
[Emek,Fraigniaud,Korman,Rosén, 2011]
[Böckenhauer,Hromkovič,Komm,Krug,Smula,Sprock, 2014]

- Guess the next bit in a bit string revealed in an online manner
- $\langle 0,\ 1,\ 0,\ ? \rangle$
- A linear amount advice is required to make mistakes on fewer than half of the bits.

**Theorem (Böckenhauer,Hromkovič,Komm,Krug,Smula,Sprock, 2014)**

On inputs of length $n$, any deterministic algorithm for 2-SGKH that is guaranteed to guess more than $\epsilon n$ bits correctly, for $0 < \epsilon \leq 1/2$, needs at least $(1 + (1-\epsilon)\log(1-\epsilon) + \epsilon \log(\epsilon))n = (1 - H(\epsilon))n$ bits of advice.

Note: Often used for lower bounds for online algorithms with advice.

# Lower bounds for approximation

What algorithms do our approximation lower bound results hold for?

# Lower bounds for approximation

What algorithms do our approximation lower bound results hold for?

Fixed priority algorithms:
We improve the results from [Borodin,B.,Larsen,Pankratov 2020] by a factor 2.

# Lower bounds for approximation

What algorithms do our approximation lower bound results hold for?

Fixed priority algorithms:
We improve the results from [Borodin,B.,Larsen,Pankratov 2020] by a factor 2.

Oblivious priority algorithms:
Adaptive priority algorithms where (for graphs) the priority function used to access the first input item in a new component does not depend on the advice.

# Lower bounds for approximation

What algorithms do our approximation lower bound results hold for?

Fixed priority algorithms:
We improve the results from [Borodin,B.,Larsen,Pankratov 2020] by a factor 2.

Oblivious priority algorithms:
Adaptive priority algorithms where (for graphs) the priority function used to access the first input item in a new component does not depend on the advice.

This is natural, not using the possibility of using a function of the names of the vertices.

# Lower bounds for approximation

What algorithms do our approximation lower bound results hold for?

Fixed priority algorithms:
We improve the results from [Borodin,B.,Larsen,Pankratov 2020] by a factor 2.

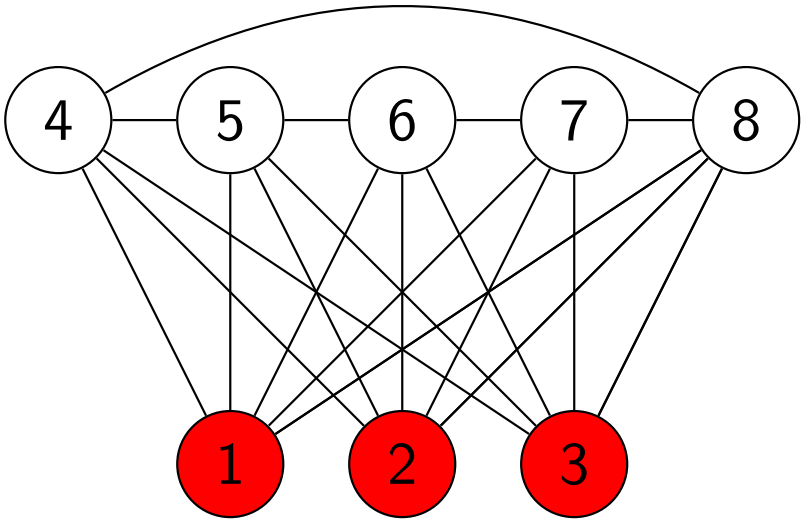
Oblivious priority algorithms:
Adaptive priority algorithms where (for graphs) the priority function used to access the first input item in a new component does not depend on the advice.

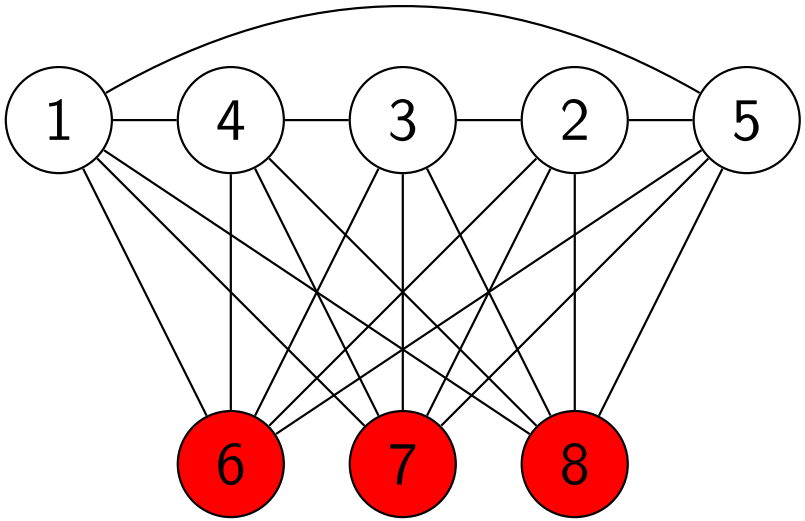
This is natural, not using the possibility of using a function of the names of the vertices.

The lower bound results for optimality also apply to fixed priority and oblivious priority algorithms.

# Another gadget pair – Independent Set example

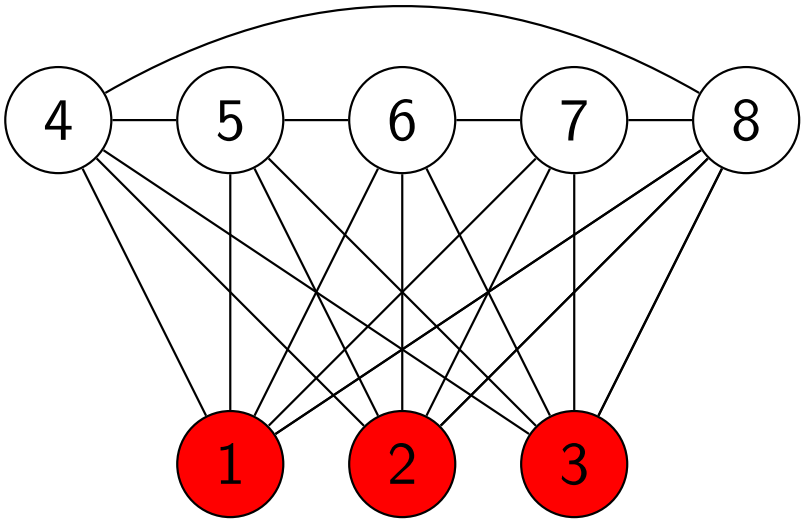
All input items are *isomorphic*.

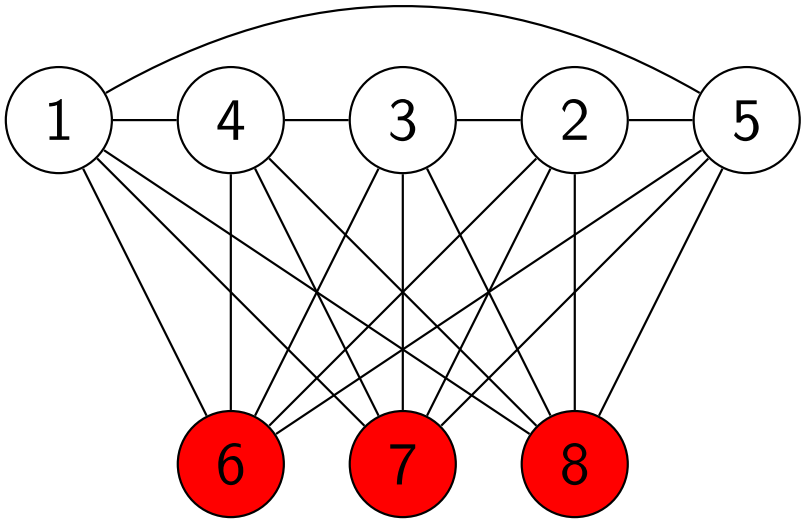


# Another gadget pair – Independent Set example

All input items are *isomorphic*.

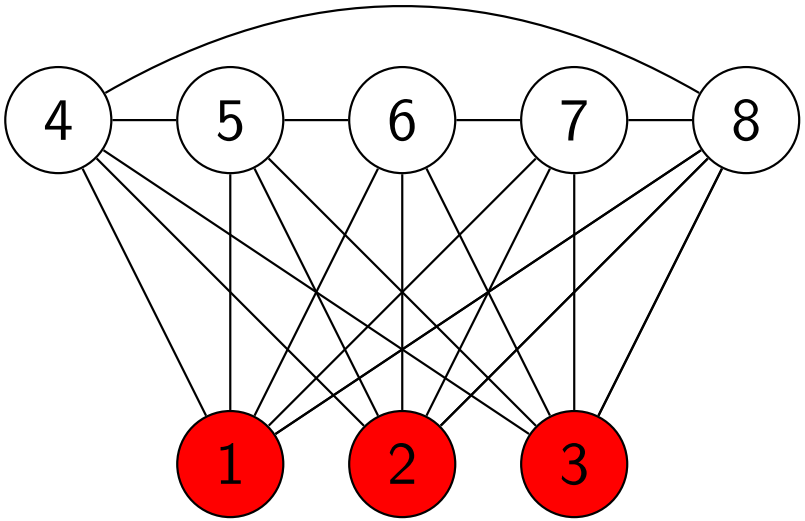


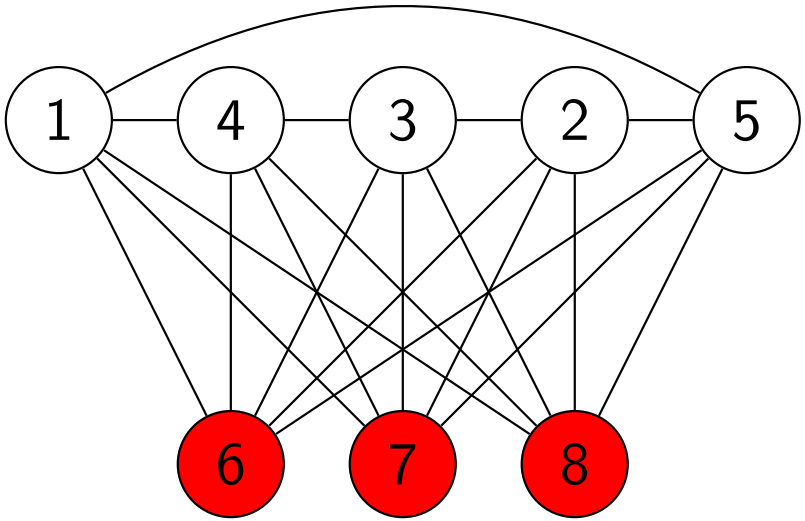




Adversary will make the first vertex selected be 1.

# Another gadget pair – Independent Set example

All input items are *isomorphic*.




Adversary will make the first vertex selected be 1.
To get an independent set of size 3, need to accept in one gadget and reject in the other.
Advantage over Vertex Cover gadget pair: Optimal solution is smaller.

# Maximization problems – lower bound

The maximization part of our general corollary:
(Let $s$ be the number of input items per gadget.)

## Theorem

For a maximization problem, if $\mathrm{OPT}(G_1) = \mathrm{OPT}(G_2) = \mathrm{BAD}(G_1) + 1 = \mathrm{BAD}(G_2) + 1$, then for any $\epsilon \in (0, 1/2]$, no oblivious priority algorithm reading fewer than $(1 - H(\epsilon))n/s$ advice bits can achieve an approximation ratio smaller than $1 + \frac{\epsilon}{OPT(G_1) - \epsilon}$.

# Maximization problems – lower bound

The maximization part of our general corollary:
(Let $s$ be the number of input items per gadget.)

**Theorem**

For a maximization problem, if $\mathrm{OPT}(G_1) = \mathrm{OPT}(G_2) = \mathrm{BAD}(G_1) + 1 = \mathrm{BAD}(G_2) + 1$, then for any $\epsilon \in (0, 1/2]$, no oblivious priority algorithm reading fewer than $(1 - H(\epsilon))n/s$ advice bits can achieve an approximation ratio smaller than $1 + \frac{\epsilon}{OPT(G_1)-\epsilon}$.

For Independent Set:

**Theorem**

For Maximum Independent Set and any $\epsilon \in (0, \frac{1}{2}]$, no oblivious priority algorithm reading fewer than $(1 - H(\epsilon))n/8$ advice bits can achieve an approximation ratio smaller than $1 + \frac{\epsilon}{3-\epsilon}$.

# Minimization problems – lower bound

The minimization part of our general corollary:
(Let $s$ be the number of input items per gadget.)

**Theorem**

For a minimization problem, if $\mathrm{OPT}(G_1) = \mathrm{OPT}(G_2) = \mathrm{BAD}(G_1)$ - 1 $= \mathrm{BAD}(G_2)$ - 1, then for any $\epsilon \in (0, 1/2]$, no oblivious priority algorithm reading fewer than $(1 - H(\epsilon))n/s$ advice bits can achieve an approximation ratio smaller than $1 + \frac{\epsilon}{OPT(G_1)}$.

# Other results

For PROBLEM and any $\epsilon \in (0, \frac{1}{2}]$, no priority algorithm reading fewer than BITS advice bits can achieve an approximation ratio smaller than RATIO.

| PROBLEM | BITS | RATIO |
|---|---|---|
| Maximum Independent Set* | $(1 - H(\epsilon))n/8$ | $1 + \frac{\epsilon}{3-\epsilon}$ |
| Maximum Independent Set** | $(1 - H(\epsilon))n/7$ | $1 + \frac{\epsilon}{4-\epsilon}$ |
| Maximum Bipartite Matching | $(1 - H(\epsilon))n/3$ | $1 + \frac{\epsilon}{3-\epsilon}$ |
| Maximum Cut | $(1 - H(\epsilon))n/8$ | $1 + \frac{\epsilon}{15-\epsilon}$ |
| Minimum Vertex Cover | $(1 - H(\epsilon))n/7$ | $1 + \frac{\epsilon}{3}$ |
| Maximum 3-Satisfiability | $(1 - H(\epsilon))n/3$ | $1 + \frac{\epsilon}{8-\epsilon}$ |
| Unit Job Scheduling with PC | $(1 - H(\epsilon))n/9$ | $1 + \frac{\epsilon}{6-\epsilon}$ |

# Comments

These results are the same as those obtained previously for fixed priority algorithms with advice, but a factor 2 better for number of bits.

# Comments

These results are the same as those obtained previously for fixed priority algorithms with advice, but a factor 2 better for number of bits.

There is a trade-off between number of bits and ratio possible for Maximum Independent Set, depending on the gadgets.

# Comments

These results are the same as those obtained previously for fixed priority algorithms with advice, but a factor 2 better for number of bits.

There is a trade-off between number of bits and ratio possible for Maximum Independent Set, depending on the gadgets.

The amount of advice required for optimality is the same as the advice required for approximation, without the $(1 - H(\epsilon))$ term.

| PROBLEM | Approximation | Optimality |
|---|---|---|
| Maximum Independent Set** | $(1 - H(\epsilon))n/7$ | $n/7$ |
| Maximum Bipartite Matching | $(1 - H(\epsilon))n/3$ | $n/3$ |
| Maximum Cut | $(1 - H(\epsilon))n/8$ | $n/8$ |
| Maximum 3-Satisfiability | $(1 - H(\epsilon))n/3$ | $n/3$ |
| Unit Job Scheduling with PC | $(1 - H(\epsilon))n/9$ | $n/9$ |

Thank you for your attention.