Exponential-Time Approximation of Hard Problems

Łukasz Kowalik
joint work with: Marek Cygan, Marcin Pilipczuk and Mateusz Wykurz

University of Warsaw, Poland

Bruxelles, 26.11.2009

Some NP-hard problems are really hard

We will focus on the following, natural problems:

- Set Cover
- Bandwidth
- Vertex Coloring
- Maximum Independent Set

Coping with NP-hardness

1. (poly-time) approximation.

Coping with NP-hardness

1. (poly-time) approximation.
   - SET COVER: no $(1-\epsilon)\log n$-approximation, unless $\mathrm{NP} \subseteq \mathrm{DTIME}(n^{\log\log n})$.
   - BANDWIDTH: no $O(1)$-approximation, unless $NP = P$
   - VERTEX COLORING: no $n^{1-\epsilon}$-approximation, unless $NP = ZPP$
   - MAXIMUM INDEPENDENT SET: no $n^{1-\epsilon}$-approximation, unless $NP = ZPP$

Coping with NP-hardness

1. (poly-time) approximation.
2. Fixed-parameter tractability

Coping with NP-hardness

1. (poly-time) approximation.
2. Fixed-parameter tractability
   - SET COVER: $W[2]$-complete.
   - BANDWIDTH: $W[t]$-hard, for any $t > 0$.
   - k-COLORING: NP-complete for any $k \geq 3$.
   - MAXIMUM INDEPENDENT SET: $W[1]$-complete

Coping with NP-hardness

1. (poly-time) approximation.
2. Fixed-parameter tractability
3. Moderately exponential-time exact algorithms

Coping with NP-hardness

1. (poly-time) approximation.
2. Fixed-parameter tractability
3. Moderately exponential-time exact algorithms
   - Set Cover: $O^*(2^m)$, $O^*(4^n)$, $O^*(2^{0.299(n+m)})$.
   - Bandwidth: $O^*(5^n)$-time and $O^*(2^n)$-space; $O^*(10^n)$ poly-space,.
   - k-coloring: $O^*(2^n)$-time and space.
   - Maximum Independent Set: $O(2^{0.276n})$-time, exp-space; $O(2^{0.288n})$-time, poly-space.

Coping with NP-hardness

1. (poly-time) approximation.
2. Fixed-parameter tractability
3. Moderately exponential-time exact algorithms
4. Moderately exponential-time approximation algorithms (our approach)

Approach One:
Reducing the Instance Size

Maximum Independent Set

Let us recall the Maximum Independent Set problem:

Instance

Undirected graph $G = (V, E)$

$I \subseteq V$ is an independent set in G when for any $x, y \in I$, $xy \notin E$.

Problem

Find the largest possible independent set in G.

Denote $n = |V|$.

Exact algorithms

- $O^*(2^{0.288n})$-time, poly space [Fomin et al. SODA'06]
- $O^*(2^{0.276n})$-time, exp space [Robson, 80-ties]

Independent Set: r-approximation, $r \in \mathbb{N}$

Input graph: $G = (V, E)$; denote $n = |V|$.

1. Partition V into r parts $V_0, \ldots, V_{r-1}$, each of size $\lceil n/r \rceil$.

Independent Set: r-approximation, $r \in \mathbb{N}$

Input graph: $G = (V, E)$; denote $n = |V|$.

1. Partition V into r parts $V_0, \ldots, V_{r-1}$, each of size $\lceil n/r \rceil$.
2. In each induced graph $G[V_i]$, find the optimal solution OPT_i in time $O(2^{0.288n/r})$.

Independent Set: r-approximation, $r \in \mathbb{N}$

Input graph: $G = (V, E)$; denote $n = |V|$.

1. Partition V into r parts $V_0, \ldots, V_{r-1}$, each of size $\lceil n/r \rceil$.
2. In each induced graph $G[V_i]$, find the optimal solution OPT_i in time $O(2^{0.288n/r})$.
3. Return the largest of OPT_i.

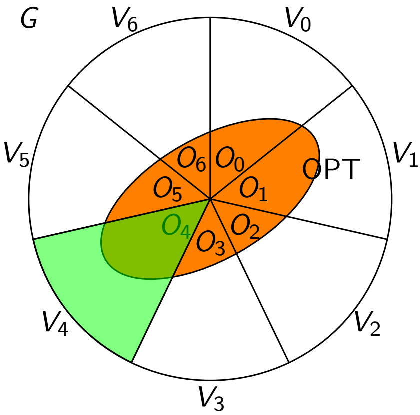
Independent Set: r-approximation, $r \in \mathbb{N}$

Input graph: $G = (V, E)$; denote $n = |V|$.

1. Partition V into r parts $V_0, \ldots, V_{r-1}$, each of size $\lceil n/r \rceil$.
2. In each induced graph $G[V_i]$, find the optimal solution OPT_i in time $O(2^{0.288n/r})$.
3. Return the largest of OPT_i.

Total time: $O(r \cdot 2^{0.288n/r}) = O^*(2^{0.288n/r})$.

Independent set – approximation guarantee



1. Recall: OPT_i = optimal solution in $G[V_i]$.
2. Let OPT be a maximum independent set in G.
3. Let $O_i = \mathrm{OPT} \cap V_i$.
4. Then for some i^*, $|O_{i^*}| \geq \mathrm{OPT}/r$.
5. Since $|\mathrm{OPT}_{i^*}| \geq |O_{i^*}|$, so $|\mathrm{OPT}_{i^*}| \geq \mathrm{OPT}/r$.

Independent Set: r-approximation, $r \in \mathbb{Q}$

Input graph: $G = (V, E)$; denote $n = |V|$, $r = p/q$.

1. Partition V into p parts $V_0, \ldots, V_{p-1}$, each of size $\lceil n/p \rceil$.

Independent Set: r-approximation, $r \in \mathbb{Q}$

Input graph: $G = (V, E)$; denote $n = |V|$, $r = p/q$.

1. Partition V into p parts $V_0, \ldots, V_{p-1}$, each of size $\lceil n/p \rceil$.
2. For $i = 0, \ldots, p-1$, let $U_i = V_i \cup V_{i+1} \cup \ldots \cup V_{i+q-1}$.
   Note: $|U_i| \leq q\lceil n/p \rceil = n/r + O(1)$

Independent Set: r-approximation, $r \in \mathbb{Q}$

Input graph: $G = (V, E)$; denote $n = |V|$, $r = p/q$.

1. Partition V into p parts $V_0, \ldots, V_{p-1}$, each of size $\lceil n/p \rceil$.
2. For $i = 0, \ldots, p-1$, let $U_i = V_i \cup V_{i+1} \cup \ldots \cup V_{i+q-1}$.
   Note: $|U_i| \leq q\lceil n/p \rceil = n/r + O(1)$
3. In each induced graph $G[U_i]$, find the optimal solution OPT_i in time $O(2^{0.288n/r})$.

Independent Set: r-approximation, $r \in \mathbb{Q}$

Input graph: $G = (V, E)$; denote $n = |V|$, $r = p/q$.

1. Partition V into p parts $V_0, \ldots, V_{p-1}$, each of size $\lceil n/p \rceil$.
2. For $i = 0, \ldots, p-1$, let $U_i = V_i \cup V_{i+1} \cup \ldots \cup V_{i+q-1}$.
   Note: $|U_i| \leq q\lceil n/p \rceil = n/r + O(1)$
3. In each induced graph $G[U_i]$, find the optimal solution OPT_i in time $O(2^{0.288n/r})$.
4. Return the largest of OPT_i.

Independent Set: r-approximation, $r \in \mathbb{Q}$

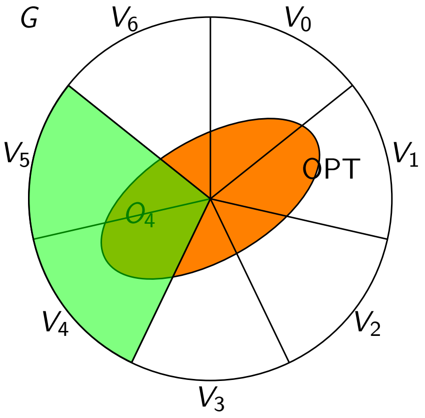
Input graph: $G = (V, E)$; denote $n = |V|$, $r = p/q$.

1. Partition V into p parts $V_0, \ldots, V_{p-1}$, each of size $\lceil n/p \rceil$.
2. For $i = 0, \ldots, p-1$, let $U_i = V_i \cup V_{i+1} \cup \ldots \cup V_{i+q-1}$.
   Note: $|U_i| \leq q\lceil n/p \rceil = n/r + O(1)$
3. In each induced graph $G[U_i]$, find the optimal solution OPT_i in time $O(2^{0.288n/r})$.
4. Return the largest of OPT_i.

Total time: $O(r \cdot 2^{0.288n/r}) = O^*(2^{0.288n/r})$.

Independent set – approximation guarantee

$r = p/q$, $p = 7$, $q = 2$.



1. Recall: OPT_i = optimal solution in $G[U_i]$.
2. Let OPT be a maximum independent set in G.
3. Let $O_i = \mathrm{OPT} \cap U_i$.
4. Then for some i^*, $|O_{i^*}| \geq \mathrm{OPT} \cdot q/p$. (Otherwise $\sum_i |O_i| < q\mathrm{OPT}$, but each element of OPT is in exactly q sets O_i, contradiction.)
5. Since $|\mathrm{OPT}_{i^*}| \geq |O_{i^*}|$, so $|\mathrm{OPT}_{i^*}| \geq \mathrm{OPT}/r$.

Assume we have an exact $O(c^n)$-time algorithm for MAXIMUM INDEPENDENT SET.

Theorem (folklore?)

For any $r \in \mathbb{Q}$ we have r-approximation in $O(C^{n/r})$ time

Reducing the instance: General approach

1. From the input instance I generate a polynomial number of smaller instances $I_1, \dots, I_k$.
2. Solve the problem exactly in each of the instances $I_1, \dots, I_k$ by an exponential time algorithm
3. Merge the solutions for $I_1, \dots, I_k$ to a solution for I.

Let us recall the Unweighted Set Cover problem:

Instance

Collection of sets $\mathcal{S} = \{S_1, \ldots, S_m\}$

The union $\bigcup \mathcal{S}$ is called the universe and denoted by U.

Problem

Find the smallest possible subcollection $\mathcal{C} \subseteq \mathcal{S}$ so that $\bigcup \mathcal{C} = U$.

Exact algorithms

- $O^*(2^m)$-time, poly space (naive)
- $O^*(2^n)$-time, poly space (Bjorklund et al FOCS'06)

WEIGHTED SET COVER

Let us recall the WEIGHTED SET COVER problem:

Instance

Collection of sets $\mathcal{S} = \{S_1, \ldots, S_m\}$
Each set has its weight $w(S_i)$.

The union $\bigcup \mathcal{S}$ is called the universe and denoted by U.

Problem

Find the lightest possible subcollection $\mathcal{C} \subseteq \mathcal{S}$ so that $\bigcup \mathcal{C} = U$.

Exact algorithms

- $O^*(2^m)$-time, poly space (naive)
- $O^*(2^n)$-time, $O(2^n)$ space (dynamic programming)
- $O^*(4^n)$-time, poly space (divide and conquer)

Unweighted Set Cover, reducing the number of sets

Approximation algorithm:

1. Join the sets of $\mathcal{S}$ into pairs:
   $S'_i = S_{2i-1} \cup S_{2i}$, for $i = 1, \ldots, m/2$ (assume m even),
   Create new instance $\mathcal{S}' = \{S'_i \mid i = 1, \ldots, m/2\}$.
2. Solve the problem for instance $\mathcal{S}'$ by the exact algorithm, in time $O(2^{m/2})$. Let $\mathcal{C}'$ be the solution.
3. Transform $\mathcal{C}'$ into a cover of $\mathcal{S}$: $\mathcal{C} = \{S_{2i-1} \cup S_{2i} \mid S'_i \in \mathcal{C}'\}$.

Unweighted Set Cover, reducing the number of sets

Approximation algorithm:

1. Join the sets of $\mathcal{S}$ into pairs:
   $S'_i = S_{2i-1} \cup S_{2i}$, for $i = 1, \ldots, m/2$ (assume m even),
   Create new instance $\mathcal{S}' = \{S'_i \mid i = 1, \ldots, m/2\}$.
2. Solve the problem for instance $\mathcal{S}'$ by the exact algorithm, in time $O(2^{m/2})$. Let $\mathcal{C}'$ be the solution.
3. Transform $\mathcal{C}'$ into a cover of $\mathcal{S}$: $\mathcal{C} = \{S_{2i-1} \cup S_{2i} \mid S'_i \in \mathcal{C}'\}$.

Proposition

This is a 2-approximation

Proof.

Let OPT be the size of the optimal cover for $\mathcal{S}$. In $\mathcal{S}'$ there is a cover of size $\leq \mathrm{OPT}$ Hence $|\mathcal{C}'| \leq \mathrm{OPT}$ and $|\mathcal{C}| \leq 2\mathrm{OPT}$. □

Unweighted Set Cover, reducing the number of sets

Approximation algorithm:

1. Join the sets of $\mathcal{S}$ into pairs:
   $S'_i = S_{2i-1} \cup S_{2i}$, for $i = 1, \ldots, m/2$ (assume m even),
   Create new instance $\mathcal{S}' = \{S'_i \mid i = 1, \ldots, m/2\}$.
2. Solve the problem for instance $\mathcal{S}'$ by the exact algorithm, in time $O(2^{m/2})$. Let $\mathcal{C}'$ be the solution.
3. Transform $\mathcal{C}'$ into a cover of $\mathcal{S}$: $\mathcal{C} = \{S_{2i-1} \cup S_{2i} \mid S'_i \in \mathcal{C}'\}$.

Question

Does it work for the weighted case?

Approximation algorithm:

1. Join the sets of $\mathcal{S}$ into pairs:
   $S'_i = S_{2i-1} \cup S_{2i}$, for $i = 1, \ldots, m/2$ (assume m even),
   Create new instance $\mathcal{S}' = \{S'_i \mid i = 1, \ldots, m/2\}$.
2. Solve the problem for instance $\mathcal{S}'$ by the exact algorithm, in time $O(2^{m/2})$. Let $\mathcal{C}'$ be the solution.
3. Transform $\mathcal{C}'$ into a cover of $\mathcal{S}$: $\mathcal{C} = \{S_{2i-1} \cup S_{2i} \mid S'_i \in \mathcal{C}'\}$.

Question

Does it work for the weighted case?

Answer

Not quite: light sets from OPT may join with heavy sets. Sorting sets ???

WEIGHTED SET COVER, reducing the number of sets

$$S_1 \leq S_2 \leq S_3 \leq S_4 \leq S_5 \leq S_6 \leq S_7 \leq S_8 \leq S_9 \leq S_{10} \leq S_{11} \leq S_{12}$$

WEIGHTED SET COVER, reducing the number of sets

$$\boxed{S_1 \leq S_2} \leq \boxed{S_3 \leq S_4} \leq \boxed{S_5 \leq S_6} \leq \boxed{S_7 \leq S_8} \leq \boxed{S_9 \leq S_{10}} \leq \boxed{S_{11} \leq S_{12}}$$

WEIGHTED SET COVER, reducing the number of sets

The sets from optimal solution are marked green.

$$\boxed{S_1 \leq S_2} \leq \boxed{S_3 \leq S_4} \leq \boxed{S_5 \leq S_6} \leq \boxed{S_7 \leq S_8} \leq \boxed{S_9 \leq S_{10}} \leq \boxed{S_{11} \leq S_{12}}$$

Weighted Set Cover, reducing the number of sets

The sets from optimal solution are marked green.
We want to show that the weight of purple pairs of sets is $\leq$ 2OPT.

$$\boxed{S_1 \leq S_2} \leq \boxed{S_3 \leq S_4} \leq \boxed{S_5 \leq S_6} \leq \boxed{S_7 \leq S_8} \leq \boxed{S_9 \leq S_{10}} \leq \boxed{S_{11} \leq S_{12}}$$

WEIGHTED SET COVER, reducing the number of sets

The sets from optimal solution are marked green.
We want to show that the weight of purple pairs of sets is $\leq 2\mathrm{OPT}$.

$$\boxed{S_1 \leq S_2} \leq \boxed{S_3 \leq S_4} \leq \boxed{S_5 \leq S_6} \leq \boxed{S_7 \leq S_8} \leq \boxed{S_9 \leq S_{10}} \leq \boxed{S_{11} \leq S_{12}}$$

Weighted Set Cover, reducing the number of sets

The sets from optimal solution are marked green.
We want to show that the weight of purple pairs of sets is $\leq 2\mathrm{OPT}$.

$$\boxed{S_1 \leq S_2} \leq \boxed{S_3 \leq S_4} \leq \boxed{S_5 \leq S_6} \leq \boxed{S_7 \leq S_8} \leq \boxed{S_9 \leq S_{10}} \leq \boxed{S_{11} \leq S_{12}}$$

Weighted Set Cover, reducing the number of sets

The sets from optimal solution are marked green.
We want to show that the weight of purple pairs of sets is $\leq 2\mathrm{OPT}$.

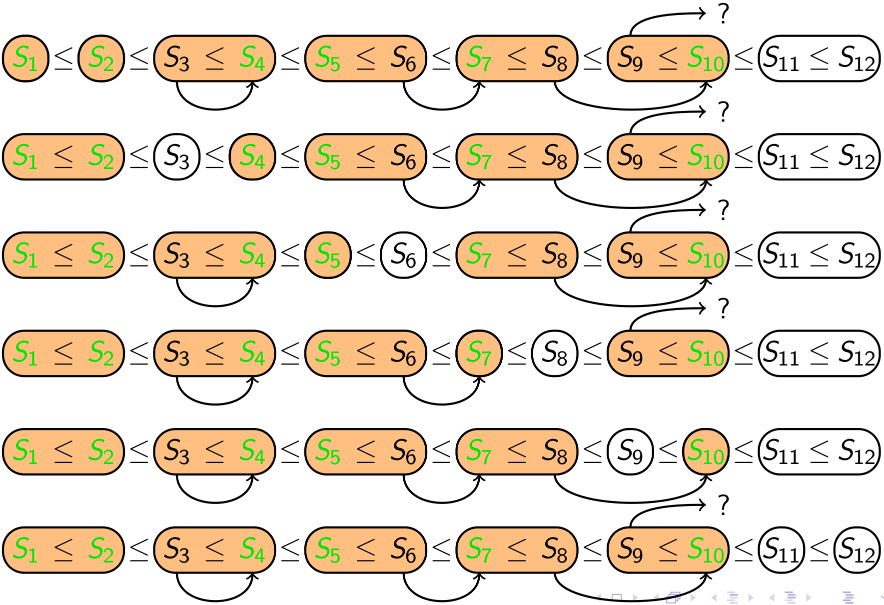
$$\boxed{S_1 \leq S_2} \leq \boxed{S_3 \leq S_4} \leq \boxed{S_5 \leq S_6} \leq \boxed{S_7 \leq S_8} \leq \boxed{S_9 \leq S_{10}} \leq \boxed{S_{11} \leq S_{12}}$$

The sets from optimal solution are marked green.
We want to show that the weight of purple pairs of sets is $\leq$ 2OPT.

?

$$(S_1 \leq S_2) \leq (S_3 \leq S_4) \leq (S_5 \leq S_6) \leq (S_7 \leq S_8) \leq (S_9 \leq S_{10}) \leq (S_{11} \leq S_{12})$$

WEIGHTED SET COVER, reducing the number of sets



Weighted Set Cover, reducing the number of sets

Assume we have an exact $O(c^m)$-time algorithm for (weighted) Set Cover.

Theorem (Cygan, K., Pilipczuk, Wykurz 2008)

There is $O^*(c^{m/r})$-time 2-approximation algorithm for (weighted) Set Cover

This trick can be generalized for any $r \in \mathbb{N}$.

Theorem (Cygan, K., Pilipczuk, Wykurz 2008)

For any $r \in \mathbb{N}$ we have r-approximation in $O^*(c^{m/r})$ time

Example 2: Set Cover, reducing the universe

Recall the standard greedy $O(\log n)$-approximation algorithm:

Greedy

1: $\mathcal{C} \leftarrow \emptyset$.
2: **while** $\mathcal{C}$ does not cover U **do**
3: Find $T \in \mathcal{S}$ so as to minimize $\frac{w(T)}{|T \setminus \bigcup \mathcal{C}|}$
4: $\mathcal{C} \leftarrow \mathcal{C} \cup \{T\}$.

Example 2: Set Cover, reducing the universe

Recall the standard greedy $O(\log n)$-approximation algorithm:

Greedy

$\mathcal{C} \leftarrow \emptyset$.
while $\mathcal{C}$ does not cover U **do**
    Find $T \in \mathcal{S}$ so as to minimize $\frac{w(T)}{|T \setminus \bigcup \mathcal{C}|}$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{T\}$.
    for each $e \in T \setminus \bigcup \mathcal{C}$ **do**
        $\mathrm{price}(e) \leftarrow \frac{w(T)}{|T \setminus \bigcup \mathcal{C}|}$

Example 2: Set Cover, reducing the universe

Recall the standard greedy $O(\log n)$-approximation algorithm:

Greedy

$\mathcal{C} \leftarrow \emptyset$.
while $\mathcal{C}$ does not cover U **do**
    Find $T \in \mathcal{S}$ so as to minimize $\frac{w(T)}{|T \setminus \bigcup \mathcal{C}|}$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{T\}$.
    for each $e \in T \setminus \bigcup \mathcal{C}$ **do**
        $\mathrm{price}(e) \leftarrow \frac{w(T)}{|T \setminus \bigcup \mathcal{C}|}$

Lemma (from the standard analysis of greedy algorithm)

Let $e_1, \ldots, e_n$ be the sequence of all elements of U in the order of covering by Greedy *(ties broken arbitrarily). Then, for each $k \in 1, \ldots, n$,* $\mathrm{price}(e_k) \leq w(\mathrm{OPT})/(n - k + 1)$

Example 2: Set Cover, reducing the universe

Lemma (from the standard analysis of greedy algorithm)

Let $e_1, \ldots, e_n$ be the sequence of all elements of U in the order of covering by Greedy *(ties broken arbitrarily). Then, for each $k \in 1, \ldots, n$,* $\mathrm{price}(e_k) \leq w(\mathrm{OPT})/(n-k+1)$

Observation

In the early phase of Greedy elements are covered cheaply.

Example 2: Set Cover, reducing the universe

Lemma (from the standard analysis of greedy algorithm)

Let $e_1, \ldots, e_n$ be the sequence of all elements of U in the order of covering by Greedy *(ties broken arbitrarily). Then, for each $k \in 1, \ldots, n$,* $\mathrm{price}(e_k) \leq w(\mathrm{OPT})/(n-k+1)$

Observation

In the early phase of Greedy elements are covered cheaply.

Exponential-Time $O(1)$-approximation

Assume we have an exact $T(n)$-time algorithm for Set Cover.

1. Run the greedy algorithm until $t \geq n/2$ elements are covered,
2. Cover the remaining elements by the exact algorithm, in time $T(n-t)$.

Example 2: Set Cover, reducing the universe

Exponential-Time $O(1)$-approximation

Assume we have an exact $T(n)$-time algorithm for Set Cover.

1. Run the greedy algorithm until $t \geq n/2$ elements are covered,
2. Cover the remaining elements by the exact algorithm, in time $T(n-t)$.

Example 2: Set Cover, reducing the universe

Exponential-Time $O(1)$-approximation

Assume we have an exact $T(n)$-time algorithm for Set Cover.

1. Run the greedy algorithm until $t \geq n/2$ elements are covered,
2. Cover the remaining elements by the exact algorithm, in time $T(n-t)$.

(Lucky) analysis

Assume we are lucky and $t = n/2$ (not bigger).

1. We pay $(H_n - H_{n/2})\mathrm{OPT} \approx (\ln n - \ln(n/2))\mathrm{OPT} = \ln 2 \cdot \mathrm{OPT}$ for the first phase,
2. we pay $\leq \mathrm{OPT}$ for the second phase.

Together we get $(1 + \ln 2)\mathrm{OPT}$.

Example 2: Set Cover, reducing the universe

Exponential-Time $O(1)$-approximation

Assume we have an exact $T(n)$-time algorithm for Set Cover.

1. Run the greedy algorithm until $t \geq n/2$ elements are covered,
2. Cover the remaining elements by the exact algorithm, in time $T(n-t)$.

Analysis

1. We pay $\leq (H_n - H_{n/2})\mathrm{OPT} \approx \ln 2 \cdot \mathrm{OPT}$ for the elements covered in phase 1, excluding the last set (that covers $e_{n/2}$),
2. We pay $\leq \mathrm{OPT}$ for the set that covers $e_{n/2}$,
3. we pay $\leq \mathrm{OPT}$ for the second phase.

Together we get $(2 + \ln 2)\mathrm{OPT}$.

Example 2: Set Cover, reducing the universe

Exponential-Time $(\ln 2 + 2)$-approximation

Assume we have an exact $T(n)$-time algorithm for Set Cover.

1. Run the greedy algorithm until $t \geq n/2$ elements are covered,
2. Cover the remaining elements by the exact algorithm, in time $T(n-t)$.

Analysis

1. We pay $\leq (H_n - H_{n/2})\mathrm{OPT} \approx \ln 2 \cdot \mathrm{OPT}$ for the elements covered in phase 1, excluding the last set (that covers $e_{n/2}$),
2. We pay $\leq \mathrm{OPT}$ for the set that covers $e_{n/2}$,
3. we pay $\leq \mathrm{OPT}$ for the second phase.

Together we get $(2 + \ln 2)\mathrm{OPT}$.

Example 2: Set Cover, reducing the universe

Exponential-Time $(\ln r + 2)$-approximation

Assume we have an exact $T(n)$-time algorithm for Set Cover.

1. Run Greedy until there are $\leq n/r$ elements not covered,
2. Cover the remaining elements by the exact algorithm, in time $T(n/r)$.

Remark 1

By stopping the Greedy algorithm when there are $\leq n/r$ uncovered elements, we get $(\ln r + 2)$-approximation in $T(n/r)$ time.

Remark 2

We show an improved algorithm with $(\ln r + 1)$-approximation in $m \times T(n/r)$ time.

WEIGHTED SET COVER, reducing the universe

Theorem (Cygan, K., Pilipczuk, Wykurz 2008)

Assume we have an exact $O(c^n)$-time algorithm for (weighted) SET COVER. For any $r \in \mathbb{Q}$ there is a $(\ln r + 1)$-approximation algorithm in $O^*(c^{n/r})$ time

Vertex Coloring

Let us recall the Vertex Coloring problem:

Instance

Undirected graph $G = (V, E)$

Problem

Find a partition of V into smallest possible number of independent sets.

Exact algorithms

- $O^*(2^n)$-time, $O(2^n)$ space (Bjorklund et al. FOCS'06)
- $O^*(2^{1.167n})$-time, poly space (Bjorklund et al. FOCS'06)

Vertex Coloring: Reducing the number of vertices

Approximation algorithm (Bjorklund, Husfeldt 2006)

1. Repeat k times (we will choose k later):
   - Remove the largest independent set I from G in $O(2^{0.288n})$-time.
   - In the original graph G_0 color all vertices in I by a new color.
2. Color vertices of the remaining graph G' by an exact algorithm.

Approximation algorithm (Bjorklund, Husfeldt 2006)

1. Repeat k times (we will choose k later):
   - Remove the largest independent set I from G in $O(2^{0.288n})$-time.
   - In the original graph G_0 color all vertices in I by a new color.
2. Color vertices of the remaining graph G' by an exact algorithm.

Let χ denote the optimum number of colors for the input graph G_0.

- In each iteration G is a subgraph of G_0, so G is χ-colorable, and hence $|I| \geq \frac{|V(G)|}{\chi}$.
- It follows that in each iteration the number of vertices decreases by a factor of $(1 - \frac{1}{\chi})$.
- $|V(G')| \leq (1 - \frac{1}{\chi})^k n \leq e^{-k/\chi} n$ and we used $k + \chi$ colors.
- Put $k = \lceil \ln r \cdot \chi \rceil$.
- Then $|V(G')| \leq n/r$, and we used $\lceil (1 + \ln r)\chi \rceil$ colors.

Our results via instance reduction

Let $T^*(n)$ denote the time of the relevant exact algorithm, up to a polynomial factor.

1. MAXIMUM INDEPENDENT SET:
   - r-approximation in $T^*(n/r)$-time.

Our results via instance reduction

Let $T^*(n)$ denote the time of the relevant exact algorithm, up to a polynomial factor.

1. Maximum Independent Set:
   - r-approximation in $T^*(n/r)$-time.
2. (Weighted) Set Cover:
   - r-approximation in $T^*(m/r)$ time,
   - $(1 + \ln r)$-approximation in $T^*(n/r)$ time.

Our results via instance reduction

Let $T^*(n)$ denote the time of the relevant exact algorithm, up to a polynomial factor.

1. Maximum Independent Set:
   - r-approximation in $T^*(n/r)$-time.
2. (Weighted) Set Cover:
   - r-approximation in $T^*(m/r)$ time,
   - $(1 + \ln r)$-approximation in $T^*(n/r)$ time.
3. Vertex Coloring:
   - Björklund & Husfeldt:
     $(1 + \ln r)$-approximation in $\max\{T^*(n/r), O^*(2^{0.288n})\}$-time.
   - $(1 + 0.247r \ln r)$-approximation in $T^*(n/r)$-time
     (best for $r \in [4.05, 58)$).
   - r-approximation in $T^*(n/r)$-time
     (best for $r \geq 58$).

Our results via instance reduction

Let $T^*(n)$ denote the time of the relevant exact algorithm, up to a polynomial factor.

1. Maximum Independent Set:
   - r-approximation in $T^*(n/r)$-time.
2. (Weighted) Set Cover:
   - r-approximation in $T^*(m/r)$ time,
   - $(1 + \ln r)$-approximation in $T^*(n/r)$ time.
3. Vertex Coloring:
   - Björklund & Husfeldt:
     $(1 + \ln r)$-approximation in $\max\{T^*(n/r), O^*(2^{0.288n})\}$-time.
   - $(1 + 0.247r \ln r)$-approximation in $T^*(n/r)$-time
     (best for $r \in [4.05, 58)$).
   - r-approximation in $T^*(n/r)$-time
     (best for $r \geq 58$).
4. Bandwidth:
   - 9-approximation in $T^*(n/2)$ time.

Our results via instance reduction

Let $T^*(n)$ denote the time of the relevant exact algorithm, up to a polynomial factor.

1. Maximum Independent Set:
   - r-approximation in $T^*(n/r)$-time.
2. (Weighted) Set Cover:
   - r-approximation in $T^*(m/r)$ time,
   - $(1 + \ln r)$-approximation in $T^*(n/r)$ time.
3. Vertex Coloring:
   - Björklund & Husfeldt:
     $(1 + \ln r)$-approximation in $\max\{T^*(n/r), O^*(2^{0.288n})\}$-time.
   - $(1 + 0.247r \ln r)$-approximation in $T^*(n/r)$-time
     (best for $r \in [4.05, 58)$).
   - r-approximation in $T^*(n/r)$-time
     (best for $r \geq 58$).
4. Bandwidth:
   - 9-approximation in $T^*(n/2)$ time.
5. Asymmetric TSP with Triangle Inequality:
   - $(1 + \log_2 r)$-approximation in $O^*(2^{n/r})$ time and space.

Reducing the instance: Summary

- If faster exact algorithm appears, immediately we have faster approximation.
- Approximation via instance reduction extends the applicability of (exact) exponential-time algorithms:

  Don't have enough time for running your algorithm for $n = 200$? Get approximate solution.

Reducing the instance: Open Problems

- For COLORING, in exponential time you can reduce the instance r times and get $(\ln r + 1)$-approximation (Björklund and Husfeldt). Can you do it for INDEPENDENT SET?
- Can *reduction of the instance size* be applied to BANDWIDTH? (Yes, but we have 9-approximation for reducing the graph by a half.)

Some reading...

- Cygan, Kowalik, Pilipczuk, Wykurz, *Exponential-Time Approximation of Hard Problems*, arXiv.
- Cygan, Kowalik, Wykurz, *Exponential-Time Approximation of Weighted Set Cover*, IPL 2009.
- Bourgeois, Escoffier, Paschos, *Efficient Approximation of Min Set Cover by Moderately Exponential Time Algorithms*, Theor. Comp. Sci. 2009.
- Bourgeois, Escoffier, Paschos, *Approximation of Min Coloring by Moderately Exponential Time Algorithms*, IPL 2009.

Approach Two:
Cutting the Search Tree

The BANDWIDTH problem

INPUT: Graph $G = (V, E)$, integer b.
PROBLEM: Find an ordering of vertices

$$\pi : V \to \{1, \ldots, n\},$$

such that "edges have length at most b", i.e.

$$\text{for every } uv \in E,\ |\pi(u) - \pi(v)| \leq b.$$

Our results: Bandwidth

- 3/2-approximation in $O^*(5^n)$ time (poly-space),
- 2-approximation in $O^*(3^n)$ time (poly-space),
- Main result: $(4r-1)$-approximation in $O^*(2^{n/r})$ time (poly-space).

Warm-up: 2-approximation in $O^*(3^n)$ time

(Inspired the exact $O(10^n)$-time algorithm by Feige and Kilian.)

Assume the bandwidth is b (we don't know it but we can run the algorithm $\log n$ times to find the smallest b for which it returns a solution).

1. Divide $\{1,\ldots,n\}$ into $\lceil n/b \rceil$ intervals of length b:
   $I_j = \{jb+1, jb+2, \ldots, (j+1)b\} \cap \{1,\ldots,n\}$.
2. Find an assignment of vertices to intervals such that
   - each interval I_j is assigned $|I_j|$ vertices,
   - adjacent vertices are assigned to the same interval or to neighboring intervals.

Warm-up: 2-approximation in $O^*(3^n)$ time

procedure $\textsc{GenerateAssignments}(A)$
    if for all j, $|A^{-1}(j)| = |I_j|$ **then**
        return A
    if for some j, $|A^{-1}(j)| > |I_j|$ **then**
        return
    else
        $v \leftarrow$ a vertex with a neighbor w already assigned.
        if $A(w) > 0$ **then**
            $\textsc{GenerateAssignments}(A \cup \{(v, A(w) - 1)\}$
        $\textsc{GenerateAssignments}(A \cup \{(v, A(w))\})$
        if $A(w) < \lceil n/b \rceil - 1$ **then**
            $\textsc{GenerateAssignments}(A \cup \{(v, A(w) + 1)\}$

procedure $\textsc{Main}$
    for $j \leftarrow 0$ **to** $\lceil n/b \rceil - 1$ **do**
        $\textsc{GenerateAssignments}\ (\{(r, j)\})$

Warm-up: 2-approximation in $O^*(3^n)$ time

1. Divide $\{1, \ldots, n\}$ into $\lceil n/b \rceil$ intervals of length b:
   $I_j = \{jb+1, jb+2, \ldots, (j+1)b\} \cap \{1, \ldots, n\}$.
2. Find an assignment of vertices to intervals such that
   - Each interval I_j is assigned $|I_j|$ vertices,
   - Adjacent vertices are assigned to the same interval or to neighboring intervals.
3. Order the vertices in each interval arbitrarily.

3-approximation in $O^*(2^n)$ time

Definition

Let A be an assignment of vertices to intervals. If one can order the vertices in each interval to get an ordering π, we say π is consistent with A.

Algorithm

1. Divide $\{1, \ldots, n\}$ into $\lceil n/b \rceil$ intervals of length $2b$:
   $I_j = \{jb+1, jb+2, \ldots, (j+2)b\} \cap \{1, \ldots, n\}$.
   (Note that intervals overlap.)
2. Generate a set of $O(n \cdot 2^n)$ assignments of vertices to intervals so that if the bandwith is b, then at least one of the assignments is consistent with an ordering of bandwidth b.
3. ... (to be continued) ...

3-approximation in $O^*(2^n)$ time

procedure GenerateAssignments(A)
    if all vertices are assigned **then**
        "Test(A)"
    else
        $v \leftarrow$ a vertex with a neighbor w already assigned.
        if $A(w) > 0$ **then**
            GenerateAssignments($A \cup \{(v, A(w) - 1)\}$)
        if $A(w) < \lceil n/b \rceil - 1$ **then**
            GenerateAssignments($A \cup \{(v, A(w) + 1)\}$)

procedure Main
    for $j \leftarrow 0$ **to** $\lceil n/b \rceil - 1$ **do**
        GenerateAssignments ($\{(r, j)\}$)

3-approximation in $O^*(2^n)$ time

Lemma („Testing A")

Let $A : V \to 2^{\{1,\dots,n\}}$ be an assignment of vertices to the intervals of size $2b$. Then there is a polynomial time algorithm such that if there is an ordering π^* of bandwidth b consistent with A, the algorithm finds an ordering π of bandwidth $3b$ consistent with A.

Proof.

1. For every edge uv, if $\max A(u) = \min A(v) - 1$, then:
   - if $|A(u)| = 2b$, replace $A(u)$ by its right half,
   - if $|A(v)| = 2b$, replace $A(v)$ by its left half.
   - (Note that π^* is still consistent with A.)
2. (now, for every edge uv, $|\max A(u) - \min A(v)| \leq 3b$)
3. Perform the standard greedy scheduling algorithm to find any ordering π consistent with A.

□

3-approximation in $O^*(2^n)$ time

Algorithm

1. Divide $\{1, \dots, n\}$ into $\lceil n/b \rceil$ intervals of length $2b$:
   $I_j = \{jb + 1, jb + 2, \dots, (j + 2)b\} \cap \{1, \dots, n\}$.
   (Note that intervals overlap.)
2. Generate a set of $O(n \cdot 2^n)$ assignments of vertices to intervals so that if the bandwith is b, then at least one of the assignments is consistent with an ordering of bandwidth b.
3. Apply the lemma to each of the assignments.

Approximation scheme

Theorem

For any $r \in \mathbb{N}$, there is a $(4r-1)$-approximation algorithm in $O^*(2^{n/r})$ time.

(Details skipped here)

Some reading...

- Cygan, Pilipczuk, *Exact and Approximate Bandwidth*, ICALP 2009.
- Furer, Gaspers, Kasiviswantathan *An Exponential-Time 2-Approximation Algorithm for Bandwidth*, IWPEC 2009.

Thank you for your attention!