Fixed-parameter tractability and approximation via linear programming

Panos Giannopoulos

Habilitationsvotrag
FU Berlin

25 October 2013

Outline

- Basics: Dealing with hard problems
- The usual suspect: Vertex Cover
- An interesting case: Independent set of (pseudo-) disks
- A good problem: Scheduling on identical machines

Dealing with NP-hard problems

Approach I: Approximation Algorithms

'...sucrifice optimality for polynomiality...'

Let Π be an optimization problem (with an objective function). Let A be an algorithm that returns, for every instance I, a feasible solution. Let $\alpha : \mathbb{Z}^+ \to \mathbb{Q}^+$.

A is a α-approximation algorithm for Π if

- there exists a polynomial p such that for every instance I, the runtime of A on I is bounded by $p(|I|)$,
- for every instance I, A produces a solution whose value is within a factor of $\alpha(|I|)$ of the value of an optimal solution for I. ($\alpha > 1$ for min, $\alpha < 1$ for max)

Approach I: Approximation Algorithms

'...sucrifice optimality for polynomiality...'
'...but get as close as you want...?'

Polynomial-time approximation scheme (PTAS):

A family of algorithms $\{A_\epsilon\}$, such that, for each $\epsilon > 0$, A_ϵ is a $(1+\epsilon)$-approximation algorithm (for min) or a $(1-\epsilon)$-approximation algorithm (for max).

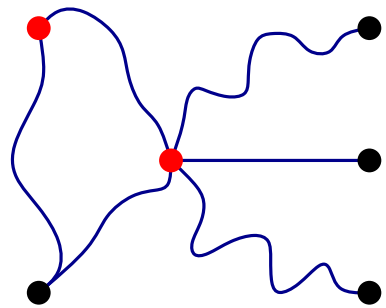
Instance I and $\epsilon > 0$:

PTAS: running time is $|I|^{f(1/\epsilon)}$
EPTAS (Efficient PTAS): running time is $f(1/\epsilon) \cdot |I|^{O(1)}$
FPTAS (Fully PTAS): running time is $(1/\epsilon)^{O(1)} \cdot |I|^{O(1)}$

Approach I: Approximation Algorithms

Example: Minimum Vertex Cover

Given a graph $G(V, E)$, find a minimum-size vertex cover, i.e., a set $V' \subseteq V$ such that every edge in E is incident to at least one vertex in V'.



- 2-approximable [Nemhauser and Trotter '75], and many others..
- no α-approximation algorithm exists for $\alpha < 1.36$..
  (unless P=NP) [Dinur and Safra '02]
- EPTAS for planar graphs [Baker '94]

Approach II: Fixed-parameter tractability

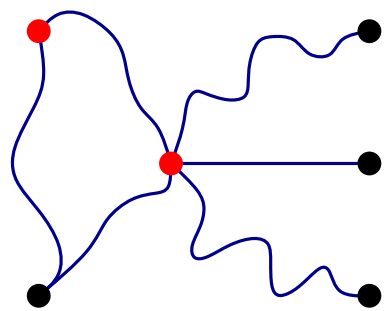
'...aim for exact solution...
...express the running time as a function of the input size and some parameter of the input...confine exponentiality in the parameter...'

Approach II: Fixed-parameter tractability

'...aim for exact solution...
...express the running time as a function of the input size and some parameter of the input...confine exponentiality in the parameter...'

Vertex Cover (decision version)

Given a graph G with n vertices, and an integer k, decide whether G has a vertex cover of size k.



Solvable

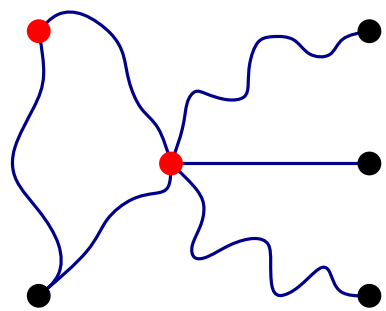
- in $O(kn^{k+1})$ time (complete enumeration)

Approach II: Fixed-parameter tractability

'...aim for exact solution...
...express the running time as a function of the input size and some parameter of the input...confine exponentiality in the parameter...'

Vertex Cover (decision version)

Given a graph G with n vertices, and an integer k, decide whether G has a vertex cover of size k.



Solvable

- in $O(kn^{k+1})$ time (complete enumeration)
- but also in $O(2^k \cdot kn)$ time: fixed-parameter tractable w.r.t k (...and also in $O(1.2738^k + kn)$ time[Chen, Kanj, Xia '10])

Approach II: Fixed-parameter tractability

A parameterization of a decision problem Π is a function that assigns an integer parameter k to each instance I of Π.

A parameter can be

- explicit in the input (e.g., the size of the cover k in the input (G, k) of Vertex Cover)
- implicit in the input (e.g., the maximum degree of the input graph G)

A parameterized problem is *fixed-parameter tractable* if it can be solved in $f(k) \cdot |I|^{O(1)}$ time, where f is a computable function depending only on k.

FPT: The class of all fixed-parameter tractable problems

Approach II: Fixed-parameter tractability

The *standard parameterization* of an optimization problem is the 'natural' parameterization (i.e., with respect to the solution size) of the corresponding decision problem.

Vertex Cover

Input	:	(G, k)
Parameter	:	k
Question	:	Is there a vertex cover of size k ?

Fact: If an optimization problem has an EPTAS then its standard parameterization is in FPT [Bazgan '95]

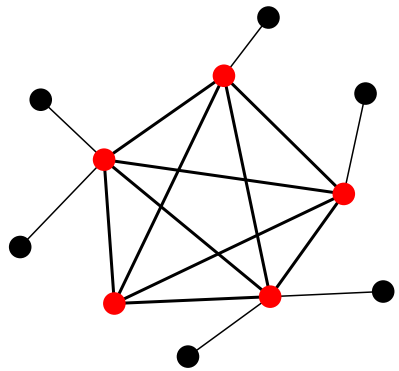
(The converse is not true, e.g., Vertex Cover)

Approach II: Fixed-parameter (in)tractability

'...are there problems for which there is evidence that they are not in FPT...?'

CLIQUE

Input : (G, k)
Parameter : k
Question : Is there a clique of size k ?
(clique: a complete subgraph)



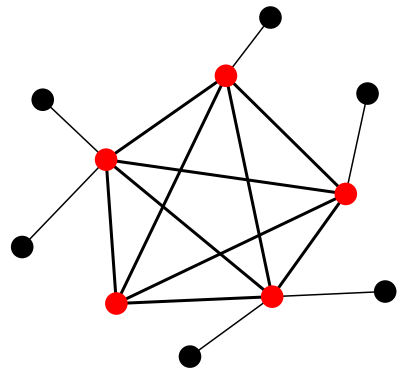
No $f(k) \cdot |G|^{O(1)}$-time algorithm is known

Approach II: Fixed-parameter (in)tractability

'...are there problems for which there is evidence that they are not in FPT...?'

CLIQUE

Input : (G, k)
Parameter : k
Question : Is there a clique of size k ?
(clique: a complete subgraph)



No $f(k) \cdot |G|^{O(1)}$-time algorithm is known

CLIQUE is W[1]-complete [Downey and Fellows '95]

Simply put: CLIQUE is not in FPT, under standard assumptions

Also: A problem that is W[1]-hard has no EPTAS

Tools: Linear and Integer Linear Programming

LP:

$$\begin{aligned} \text{min} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \\ \text{where} \quad & x \in \mathbb{R}^n,\ A \in \mathbb{Q}^{m \times n},\ b \in \mathbb{Q}^m,\ c \in \mathbb{Q}^n \end{aligned}$$

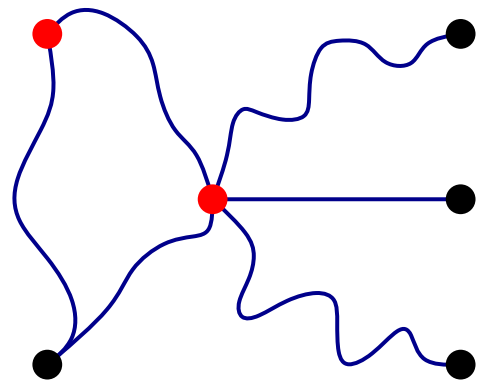
Fact: n-variable LP can be solved in $O(n^3 L)$ arithmetic operations and space polynomial in L, where L is the input length.
[Karmarkar '84, Ye '91]

ILP-F:

$$\begin{aligned} \text{Input} \quad & : A \in \mathbb{Z}^{m \times n},\ b \in \mathbb{Z}^m \\ \text{Parameter} \quad & : n \ (\#\ \text{variables}) \\ \text{Question} \quad & : \text{Is there } x \in \mathbb{Z}^n \text{ such that } Ax \leq b\ ? \end{aligned}$$

Fact: n-variable ILP-F can be solved in $O(n^{2.5n+o(n)} L)$ arithmetic operations and space polynomial in L, where L is the input length.
[Lenstra '83, Kannan '87, Frank and Tardos '87]

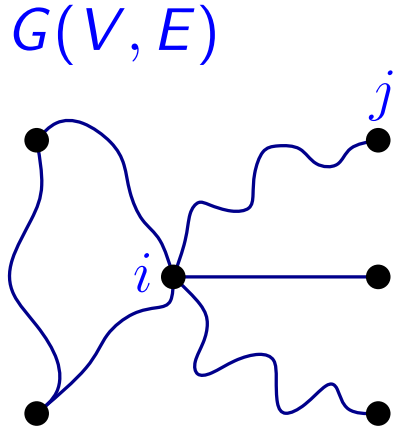
Vertex Cover



ILP...

ILP:

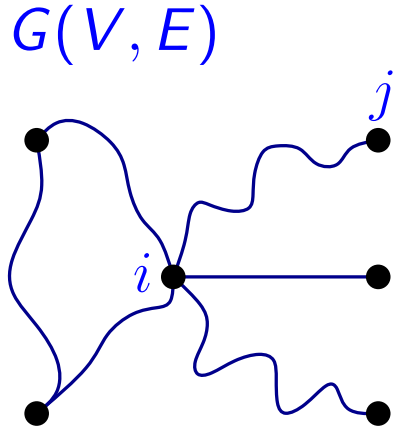
$$\begin{aligned} \text{min} \quad & \sum_{i \in V} x_i \\ \text{subject to} \quad & x_i + x_j \geq 1 \quad \forall ij \in E \\ & x_i \in \{0, 1\} \quad \forall i \in V \end{aligned}$$



ILP...and relaxation

LP:

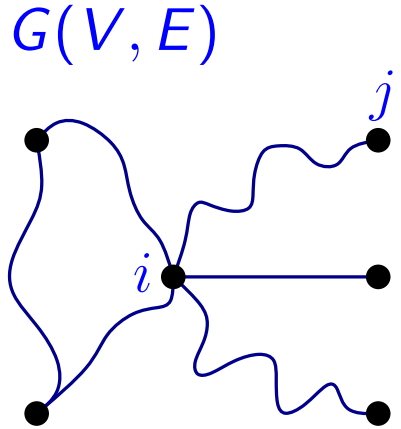
$$\begin{aligned} \text{min} \quad & \sum_{i \in V} x_i \\ \text{subject to} \quad & x_i + x_j \geq 1 \quad \forall ij \in E \\ & 0 \leq x_i (\leq 1) \quad \forall i \in V \end{aligned}$$



ILP...and relaxation

LP:

$$\begin{aligned} \text{min} \quad & \sum_{i \in V} x_i \\ \text{subject to} \quad & x_i + x_j \geq 1 \quad \forall ij \in E \\ & 0 \leq x_i (\leq 1) \quad \forall i \in V \end{aligned}$$

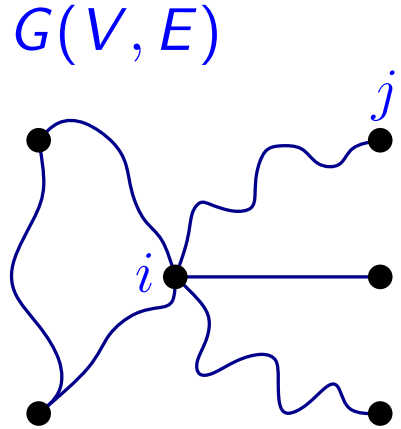


Main idea: Solve the LP, round the fractional solution to an integral solution that is feasible for ILP and whose cost is not 'much' more than the cost of LP solution

ILP...and relaxation

LP:

$$\begin{aligned} \min \quad & \sum_{i \in V} x_i \\ \text{subject to} \quad & x_i + x_j \geq 1 \quad \forall ij \in E \\ & 0 \leq x_i (\leq 1) \quad \forall i \in V \end{aligned}$$



Main idea: Solve the LP, round the fractional solution to an integral solution that is feasible for ILP and whose cost is not 'much' more than the cost of LP solution

Here: Let $\{x_i\}$ be an optimal solution for LP

Take $C = \{i \in V \mid x_i \geq 1/2\}$

Note that C is a cover with

$$|C| \leq \sum_{i \in V} 2x_i = 2OPT(LP) \leq 2OPT(ILP) \leq 2minVC$$

i.e., a 2-approximation

A combinatorial explanation for VC LP

(...or what is really happening)

$G(V, E)$

Theorem[Nemhauser and Trotter '75]:

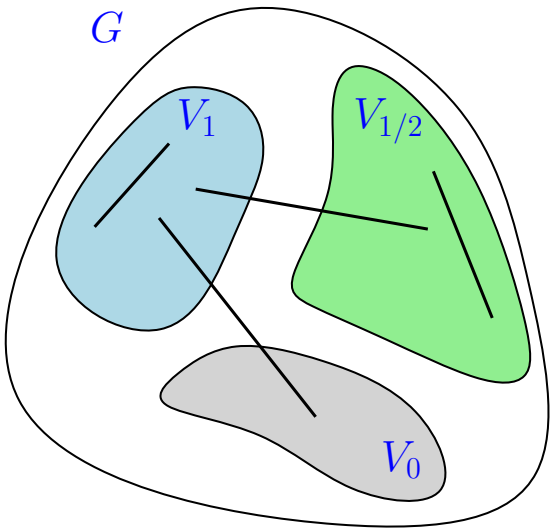
1. $\exists$ opt. sol. $\{x_i\}$ for LP that is half-integral, i.e., $x_i \in \{0, 1/2, 1\}$ (and poly-time computable).

LP:

$$\begin{aligned} \min \quad & \sum_{i \in V} x_i \\ \text{subject to} \quad & x_i + x_j \geq 1 \quad \forall ij \in E \\ & 0 \leq x_i (\leq 1) \quad \forall i \in V \end{aligned}$$

A combinatorial explanation for VC LP

(...or what is really happening)



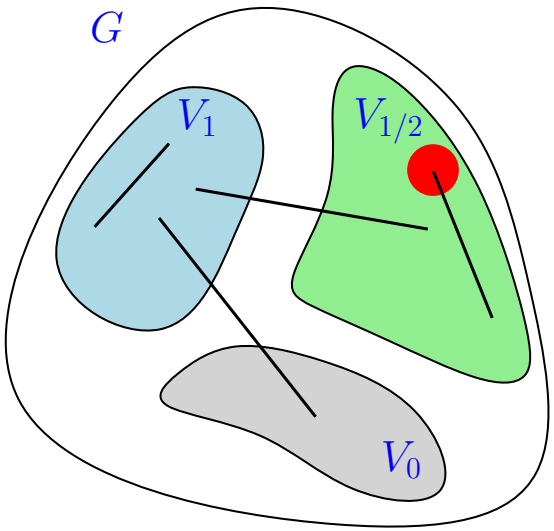
Theorem[Nemhauser and Trotter '75]:

1. $\exists$ opt. sol. $\{x_i\}$ for LP that is half-integral, i.e., $x_i \in \{0, 1/2, 1\}$ (and poly-time computable).

2. Consider the partition of V into $V_1 = \{i \in V \mid x_i = 1\}$, $V_{1/2} = \{i \in V \mid x_i = 1/2\}$, and $V_0 = \{i \in V \mid x_i = 0\}$

A combinatorial explanation for VC LP

(...or what is really happening)



Theorem[Nemhauser and Trotter '75]:

1. $\exists$ opt. sol. $\{x_i\}$ for LP that is half-integral, i.e., $x_i \in \{0, 1/2, 1\}$ (and poly-time computable).

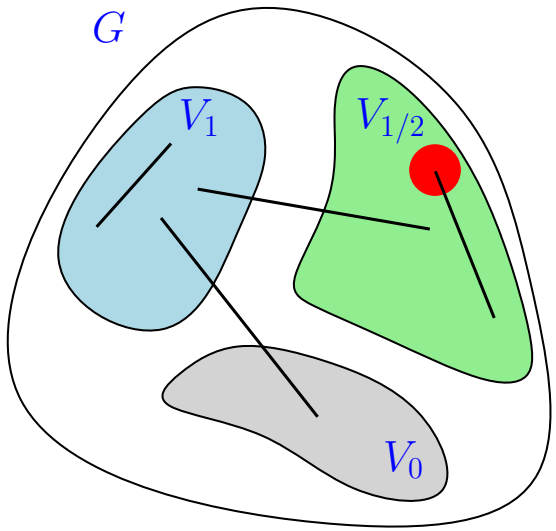
2. For the partition of V into $V_1 = \{i \in V \mid x_i = 1\}$, $V_{1/2} = \{i \in V \mid x_i = 1/2\}$, and $V_0 = \{i \in V \mid x_i = 0\}$ we have

(i) $\exists$ min.VC of G that consists of V_1 and of a min.VC of $G[V_{1/2}]$ i.e., $minVC(G[V_{1/2}]) + |V_1| = minVC(G)$

(ii) $minVC(G[V_{1/2}]) \geq |V_{1/2}|/2$

A combinatorial explanation for VC LP

(...or what is really happening)



Theorem[Nemhauser and Trotter '75]:

1. $\exists$ opt. sol. $\{x_i\}$ for LP that is half-integral, i.e., $x_i \in \{0, 1/2, 1\}$ (and poly-time computable).

2. For the partition of V into $V_1 = \{i \in V \mid x_i = 1\}$, $V_{1/2} = \{i \in V \mid x_i = 1/2\}$, and $V_0 = \{i \in V \mid x_i = 0\}$ we have

(i) $\exists$ min.VC of G that consists of V_1 and of a min.VC of $G[V_{1/2}]$ i.e., $minVC(G[V_{1/2}]) + |V_1| = minVC(G)$

(ii) $minVC(G[V_{1/2}]) \geq |V_{1/2}|/2$

In other words, forget V_0 and try to find a cover for $G[V_{1/2}]$

(Note: $V_1 \cup V_{1/2}$ gives a 2-approximation)

and...

...off to FPTness

Task: Given k, find a VC of size k or report that none exists

Simple: Focus on $G[V_{1/2}]$, reduce its size with the NT Thm!
[Chen, Kanj, Jia '01]

...off to FPTness

Task: Given k, find a VC of size k or report that none exists

Simple: Focus on $G[V_{1/2}]$, reduce its size with the NT Thm!
[Chen, Kanj, Jia '01]

Let $k' = k - |V_1|$ ('new' parameter with reduced value)

- $k' < 0 \implies$ NO (since $minVC \geq |V_1|$)
- $k' = 0 \implies$
  - NO, if $G[V_{1/2}]$ has at least one edge
  - YES, otherwise (V_1 is a cover)
- $k' > 0$ and $|V_{1/2}| > 2k' \implies$ NO (since $minVC(G[V_{1/2}]) > k'$)

...off to FPTness

Task: Given k, find a VC of size k or report that none exists

Simple: Focus on $G[V_{1/2}]$, reduce its size with the NT Thm!
[Chen, Kanj, Jia '01]

Let $k' = k - |V_1|$ ('new' parameter with reduced value)

- $k' < 0 \implies$ NO (since $minVC \geq |V_1|$)
- $k' = 0 \implies$
  - NO, if $G[V_{1/2}]$ has at least one edge
  - YES, otherwise (V_1 is a cover)
- $k' > 0$ and $|V_{1/2}| > 2k' \implies$ NO (since $minVC(G[V_{1/2}]) > k'$)
- **other:** 'new' instance (kernel) $(G[V_{1/2}], k')$ with $|V_{1/2}| \leq 2k'$ s.t.

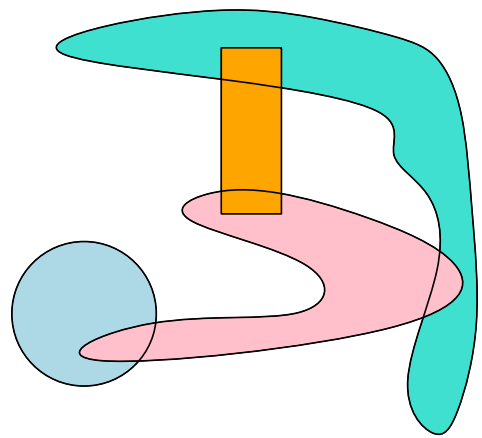
  G has a VC of size $k \iff G[V_{1/2}]$ has a VC of size k'

  Apply brute-force or an fpt-algorithm for Vertex Cover on $(G[V_{1/2}], k')$

More applications...

- LP-based Apx: > 100 papers (rand. rounding, primal-dual,...)
- $(2 - \frac{\log\log n}{2\log n})$-apx for Min. VC [Yehuda and Even '85]
- PTAS for Generalized VC (ext. of NT Thm.)
  [Yehuda and Hermelin and Rawitz '10]
- Kernel for Bounded Degree Deletion (ext. of NT Thm.)
  [Fellows, Guo, Moser, Niedermeier '09]
- Above Guarantee Vertex Cover
  (param.$=k - |maxMaching|$)(NT Thm. + preprocess. + branching)
  [Lokshtanov, Narayanaswamy, Raman, Ramanujan, Saurabh '11]

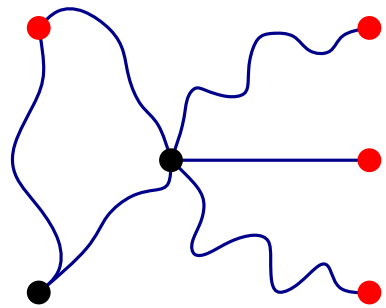
Independent set of (pseudo-) disks



Independent set in general

Maximum Independent Set

Given a graph $G(V, E)$, find a maximum-size independent set, i.e., a set $V' \subseteq V$ such that no two vertices in V' are connected by an edge.

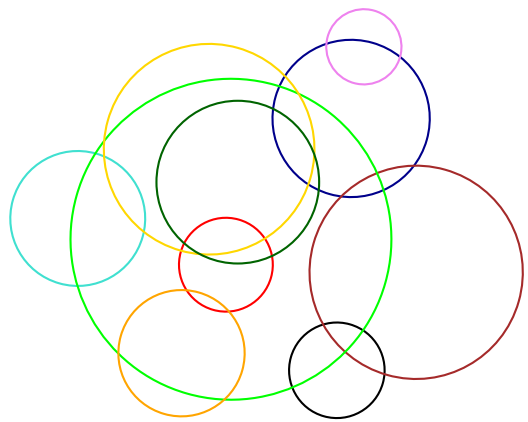


- no $O(n^{\epsilon-1})$-approximation algorithm, for any $\epsilon > 0$ (unless P=NP) [Håstad '99]
- W[1]-complete [Downey and Fellows '95]
- EPTAS for planar graphs: $O(2^{O(1/\epsilon)}n^2)$ [Baker '94]
- FPT (stand. param.) for planar graphs: $O(2^{O(\sqrt{k})}n)$ [Fomin and Thilikos '03], [Alber, Fellows, Niedermeier '04]

Independent set of geometric objects - something in between

Problem

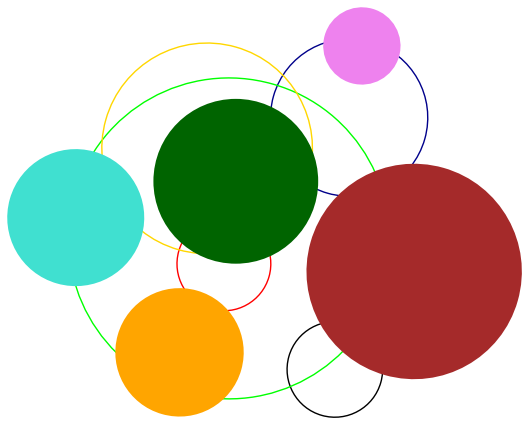
Given a set $F = \{f_1, \ldots, f_n\}$ of n objects in $\mathbb{R}^2$ find a maximum-size subset of F such that no two objects intersect.



Independent set of geometric objects - something in between

Problem

Given a set $F = \{f_1, \ldots, f_n\}$ of n objects in $\mathbb{R}^2$ find a maximum-size subset of F such that no two objects intersect.



Reduces to Maximum Independent Set on the intersection graph of F

Independent set of geometric objects - something in between

Known:

- NP-hard (axis-parallel segments, unit disks, unit squares...)
- PTAS for disks, squares [Chan '03], [Erlebach, Jansen, Seidel '05]
- PTAS for pseudo-disks [Chan and Har-Peled '12]

- $O(\log \log n)$-approximation for axis-parallel rectangles [Chalermsook and Chuzhoy '03]
- $O(\sqrt{OPT})$-approximation for segments [Agarwal and Mustafa '06]

- W[1]-hard for unit disks, unit squares, unit segments with arbitrary directions [Marx '05], [Marx '06]
- FPT for λ-precision unit disks, segments with fixed directions [Alber and Fiala'04], [Marx '06], [Kára and Kratochvíl '06]

and many others...

Independent set of geometric objects - something in between

Known:

- NP-hard (axis-parallel segments, unit disks, unit squares...)
- PTAS for disks, squares [Chan '03], [Erlebach, Jansen, Seidel '05]
- PTAS for pseudo-disks [Chan and Har-Peled '12]

$\longrightarrow$ **O(1)-approximation for pseudo-disks** (weighted as well) [Chan and Har-Peled '12]

- $O(\log \log n)$-approximation for axis-parallel rectangles [Chalermsook and Chuzhoy '03]
- $O(\sqrt{OPT})$-approximation for segments [Agarwal and Mustafa '06]

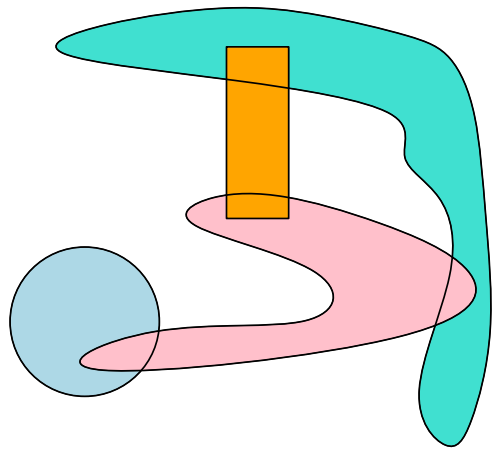
- W[1]-hard for unit disks, unit squares, unit segments with arbitrary directions [Marx '05], [Marx '06]
- FPT for λ-precision unit disks, segments with fixed directions [Alber and Fiala'04], [Marx '06], [Kára and Kratochvíl '06]

and many others...

Pseudo-disks

A set of simply connected objects (regions) in $\mathbb{R}^2$, each bounded by a simple closed (Jordan) curve, is a collection of **pseudo-disks** if the boundaries of every two objects intersect at most twice.

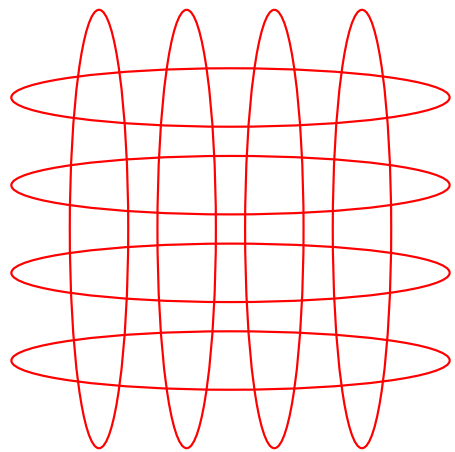
pseudo-disks:



Pseudo-disks

A set of simply connected objects (regions) in $\mathbb{R}^2$, each bounded by a simple closed (Jordan) curve, is a collection of **pseudo-disks** if the boundaries of every two objects intersect at most twice.

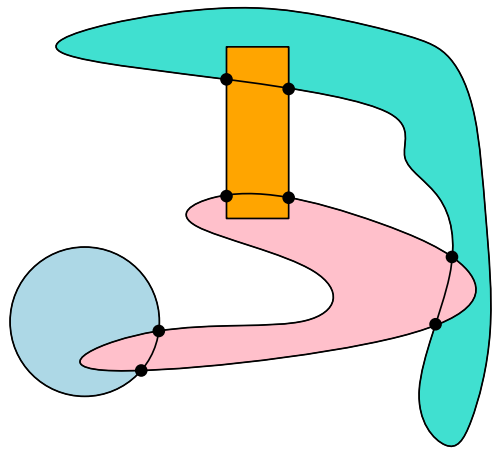
Not pseudo-disks:



Pseudo-disks

A set of simply connected objects (regions) in $\mathbb{R}^2$, each bounded by a simple closed (Jordan) curve, is a collection of **pseudo-disks** if the boundaries of every two objects intersect at most twice.

pseudo-disks:



Linear union complexity:
The complexity of the (boundary of the) union of a set of k pseudo-disks is $O(k)$. In particular, there are at most $6k$ vertices on the boundary. [Kedem, Livne, Pach, Sharir '86]

(The following hold for any set with linear union complexity)

ILP and LP relaxation

$F = \{f_1, \ldots, f_n\}$: n objects in $\mathbb{R}^2$ (no full containment)

$\mathcal{A}(F)$: arrangement induced F, $\mathcal{V}(F)$: vertices of $\mathcal{A}(F)$

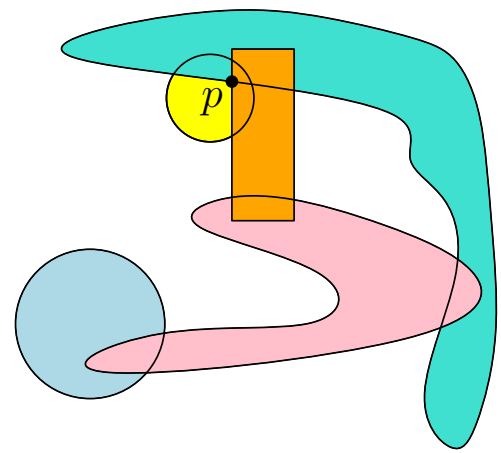
$x_i \leftarrow f_i$

LP:

$$\max \sum_{i \in [n]} x_i$$

$$\text{s. t.} \sum_{f_i \ni p} x_i \leq 1 \quad \forall \text{ vertex } p \in \mathcal{V}(F)$$

$$0 \leq x_i \leq 1 \quad \forall i \in [n]$$



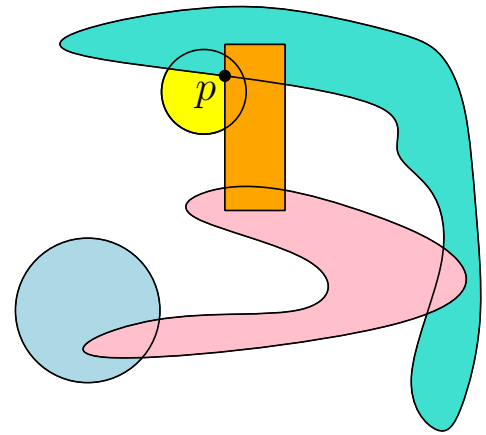
Approach

LP:

$$\max \sum_{i \in [n]} x_i$$

$$\text{s. t.} \sum_{f_i \ni p} x_i \leq 1 \quad \forall \text{ vertex } p \in \mathcal{V}(F)$$

$$0 \leq x_i \leq 1 \quad \forall i \in [n]$$



0. $R := \emptyset$.
1. Solve the LP: let $\{x_i\}$ be an optimal solution.
2. Randomly put each object f_i into R with probability x_i.

$\rightarrow$ **Show:** For the intersection graph $G(R, K)$ induced by R,
$E[|K|] = O(E[|R|])$

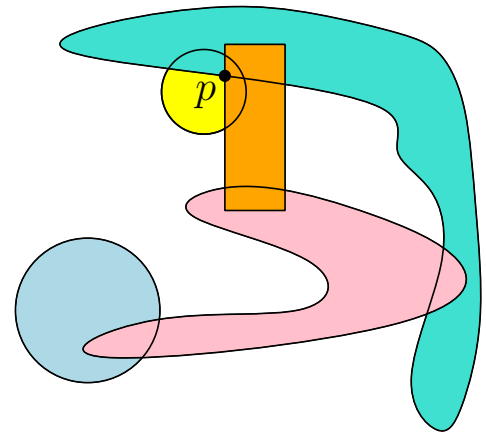
Approach

LP:

$$\max \sum_{i \in [n]} x_i$$

$$\text{s. t.} \sum_{f_i \ni p} x_i \leq 1 \quad \forall \text{ vertex } p \in \mathcal{V}(F)$$

$$0 \leq x_i \leq 1 \quad \forall i \in [n]$$



0. $R := \emptyset$.
1. Solve the LP: let $\{x_i\}$ be an optimal solution.
2. Randomly put each object f_i into R with probability x_i.

$\rightarrow$ **Show:** For the intersection graph $G(R, K)$ induced by R,
$E[|K|] = O(E[|R|])$

3. **Turán's Theorem:**
   Any graph with n vertices and average degree $\bar{d}$ has an independent set of size $\geq n/(\bar{d}+1)$.
   (e.g., use min. degree greedy algorithm)

Explanation...

$F = \{f_1, \ldots, f_n\}$: objects in $\mathbb{R}^2$ (and $R \subseteq F$: $|\mathcal{U}(R)| \leq \rho|R|$)

$\{x_i\}$: opt. sol. for LP, $G(R, K)$: induced intersection graph

Note: – $E[|R|] = \sum_{i \in [n]} x_i = OPT(LP) \geq maxIS$

– $E[|K|] = \sum_{i<j:\, f_i \cap f_j \neq \emptyset} x_i x_j$

Explanation...

$F = \{f_1, \ldots, f_n\}$: objects in $\mathbb{R}^2$ (and $R \subseteq F$: $|\mathcal{U}(R)| \leq \rho|R|$)

$\{x_i\}$: opt. sol. for LP, $G(R, K)$: induced intersection graph

Note: – $E[|R|] = \sum_{i \in [n]} x_i = OPT(LP) \geq maxIS$

– $E[|K|] = \sum_{i<j:\, f_i \cap f_j \neq \emptyset} x_i x_j$

Lemma: $\sum_{i<j:\, f_i \cap f_j \neq \emptyset} x_i x_j \leq 4\rho E[|R|]$

Thus: $E[|K|] \leq 4\rho E[|R|]$

...and using Turán's Thm. on R we get an ind. set $I \subset R$ with

$$E[|I|] \geq E\left[\frac{|R|}{2K/|R| + 1}\right] \geq \cdots \geq \frac{E[|R|]}{(2E[|K|]/E[|R|]) + 1} \geq \frac{maxIS}{8\rho + 1}$$

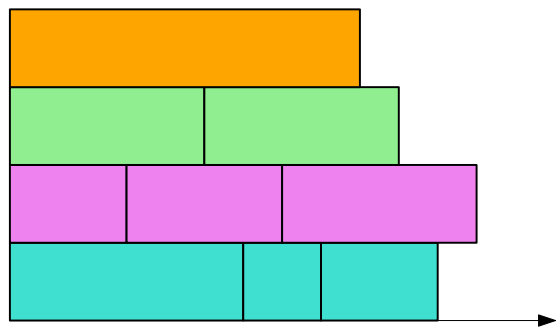
More on...

The $O(1)$-apx algorithm can be

- generalized to the weighted case
- derandomized

[Chan and Har-Peled '12]

Scheduling on identical machines



Scheduling on identical machines

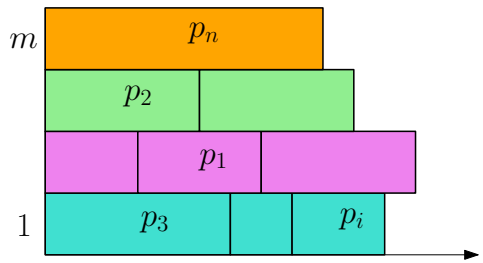
Minimum Makespan Scheduling

Given : n jobs, processing times $p_1, \ldots, p_n$, $p_j \in \mathbb{N}$
m identical machines (that run in parallel)

Compute : non-preemtive schedule that assigns each job to some machine such that the total processing time is minimized, i.e.,

a partition $S_1, \ldots, S_m$ of $[n]$ that minimizes

$$\max_{1 \le i \le m} \sum_{j \in S_i} p_j$$

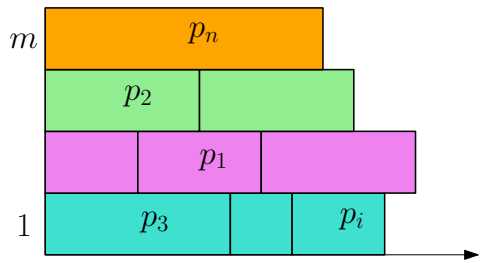


Scheduling on identical machines

Minimum Makespan Scheduling

Given : n jobs, processing times $p_1, \ldots, p_n$, $p_j \in \mathbb{N}$
m identical machines (that run in parallel)

Compute : non-preemtive schedule that assigns each job to some machine such that the total processing time is minimized, i.e.,



a partition $S_1, \ldots, S_m$ of $[n]$ that minimizes

$$\max_{1 \le i \le m} \sum_{j \in S_i} p_j$$

- $4/3$-approximation [Graham '69]
- FPT (stand. param.) [Alon, Azar, Woeginger, Yadid '98]
- EPTAS [Alon, Azar, Woeginger, Yadid '98]: uses the fpt-algorithm!

Fixed-parameter tractbility via ILP

Makespan Scheduling:

Input : $m, p_1, \ldots, p_n, k \in \mathbb{N}$
Parameter : k
Question : Is there a schedule with makespan $\leq k$?

$n = f(k)$

ILP-F:

Input : $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$
Parameter : n (# variables)
Question : Is there $x \in \mathbb{Z}^n$ such that $Ax \leq b$?

Idea: Reduce Makespan Scheduling to ILP-F with $n = f(k)$ variables and use the fact that ILP-F is fixed-parameter tractable w.r.t n

1. Characterize assignments of cost $\leq k$

Instance: $m, p_1, \ldots, p_n, k \in \mathbb{N}$
Question: Is there a schedule of cost $\leq k$?

Observe:

- $p_i \leq k$, $\forall\, i \in [n]$ (otherwise answer NO)
- assignment A for a single machine:

  $\bar{a} = (a_1, \ldots, a_k) \in \mathbb{N}^k$, with $0 \leq a_i \leq k$, $\forall\, i \in [k]$

  i.e., exactly a_i jobs of cost i are assigned to the machine
  and $cost(\bar{a}) = \sum_{i \in [k]} a_i \cdot i$
- there are at most $(k+1)^k$ assignments of cost $\leq k$

2. Model schedules as ILP

$A = \{\text{assigment } \bar{\alpha} \text{ with } cost(\bar{\alpha}) \leq k\}$, $|A| \leq (k+1)^k$

Note: A schedule can be defined by the number of machines that follow an assignment $\bar{\alpha} \in A$

- variable $x_{\bar{a}} = \#$ machines following assignment $\bar{a}$, $\forall\, \bar{a} \in A$
- $b_j = \#$ jobs $i \in [n]$ with processing time $p_i = j$, $\forall\, j \in [k]$

2. Model schedules as ILP

$A = \{\text{assigment } \bar{\alpha} \text{ with } cost(\bar{\alpha}) \leq k\}$, $|A| \leq (k+1)^k$

Note: A schedule can be defined by the number of machines that follow an assignment $\bar{\alpha} \in A$

- variable $x_{\bar{a}} = \#$ machines following assignment $\bar{a}$, $\forall\, \bar{a} \in A$
- $b_j = \#$ jobs $i \in [n]$ with processing time $p_i = j$, $\forall\, j \in [k]$

ILP:

$$\sum_{\bar{a} \in A} x_{\bar{a}} = m$$

(one assignment per machine)

$$\sum_{\bar{a}=(a_1,\ldots,a_k) \in A} a_j \cdot x_{\bar{a}} = b_j, \quad \forall j \in [k]$$

(right # jobs with processing time j is scheduled)

$$x_{\bar{a}} \in \mathbb{N}, \quad \forall \bar{a} \in A$$

(If ILP is feasible, a schedule can be easily found, e.g., by testing all possible choices for every variable)

EPTAS

Fact[Graham '66]: Greedy algorithm (*list scheduling*) gives a 2-approximation: *assign the next job to the first available machine.*

$L = \max\left(\{p_i \mid i \in [n]\} \cup \left\{\frac{1}{m}\sum_{i=1}^{n} p_i\right\}\right)$, we have $L \leq OPT$,

and the algorithm computes a feasible schedule of cost $\leq 2L$.

EPTAS

Fact[Graham '66]: Greedy algorithm (*list scheduling*) gives a 2-approximation: *assign the next job to the first available machine.*

$L = \max\left(\{p_i \mid i \in [n]\} \cup \left\{\frac{1}{m}\sum_{i=1}^{n} p_i\right\}\right)$, we have $L \leq OPT$,

and the algorithm computes a feasible schedule of cost $\leq 2L$.

Overview of EPTAS:

1. Given $k \in \mathbb{N}$, compute a schedule for the 'big' jobs ($p_i \geq L/k$) only, with cost that approximates OPT (whole instance) to a factor of $(1 + 1/k)$.

   This reduces to (the standard parameterization) Makespan Scheduling for a scaled version of the restricted instance.

EPTAS

Fact[Graham '66]: Greedy algorithm (*list scheduling*) gives a 2-approximation: *assign the next job to the first available machine.*

$L = \max\left(\{p_i \mid i \in [n]\} \cup \left\{\frac{1}{m}\sum_{i=1}^{n} p_i\right\}\right)$, we have $L \leq OPT$,

and the algorithm computes a feasible schedule of cost $\leq 2L$.

Overview of EPTAS:

1. Given $k \in \mathbb{N}$, compute a schedule for the 'big' jobs ($p_i \geq L/k$) only, with cost that approximates OPT (whole instance) to a factor of $(1 + 1/k)$.

   This reduces to (the standard parameterization) Makespan Scheduling for a scaled version of the restricted instance.

2. Schedule the 'small' ($p_i < L/k$) jobs so as not to increase the cost of by too much: Greedy will do.

More on...

Scheduling on unrelated machines
(different process. times on each machine):

- 2-apx. algorithm based on LP-relaxation
- no α-approximation algorithm exists for $\alpha < 3/2$

[Lenstra, Shmoys, Tardos '90]

Other applications of ILP to fixed-parameter tractability

- Fixed-parameter algorithms for Closest String and related problems [Gramm, Niedermeier, Rossmanith '03]
- Graph layout problems parameterized by (the size of the minimum) vertex cover
  [Fellows, Lokshtanov, Misra, Rosamond, Saurabh '08]

It's time for discussion!

Appendix

Parameterized complexity

Parameterized complexity theory provides a framework (hierarchy of classes) for showing fixed-parameter intractability

- the first level is the class W[1] (FPT$\subseteq$W[1])
- hardness is established via fpt-reductions
- a W[1]-hard problem is not in FPT, unless FPT=W[1]
- FPT=W[1] $\implies$ n-variable 3-SAT can be solved in $2^{o(n)}$ time (not believed to be true)

- Clique is W[1]-complete [Downey and Fellows '95]
- Independent Set is W[1]-complete [Downey and Fellows '95]

Scheduling big jobs...

Instance I: $m, p_1, \ldots, p_n$ with $p_1 \geq p_2 \geq \ldots \geq p_n$

$\bar{S} = (S_1, \ldots, S_m)$ an opt. schedule for I, $OPT(I) = \max_{1 \leq j \leq m} \sum_{i \in S_j} p_i$

Big jobs: jobs $i \in [n]$ with $p_i \geq L/k$

Assume: jobs $1, 2, \ldots, n'$ are big, for some $n' \in [n]$

New instance I': $m, p'_1, \ldots, p'_{n'}$ with $p'_i = \lfloor (k^2/L) \cdot p_i \rfloor$

Properties: $p'_i \geq k$ and $p_i \leq (1 + 1/k) \cdot (L/k^2) \cdot p'_i$

Lemma: $OPT(I') \leq 2k^2$.

Scheduling big jobs...

Instance I: $m, p_1, \ldots, p_n$ with $p_1 \geq p_2 \geq \ldots \geq p_n$

$\bar{S} = (S_1, \ldots, S_m)$ an opt. schedule for I, $OPT(I) = \max_{1 \leq j \leq m} \sum_{i \in S_j} p_i$

Big jobs: jobs $i \in [n]$ with $p_i \geq L/k$

Assume: jobs $1, 2, \ldots, n'$ are big, for some $n' \in [n]$

New instance I': $m, p'_1, \ldots, p'_{n'}$ with $p'_i = \lfloor (k^2/L) \cdot p_i \rfloor$

Properties: $p'_i \geq k$ and $p_i \leq (1 + 1/k) \cdot (L/k^2) \cdot p'_i$

Lemma: $OPT(I') \leq 2k^2$.

Compute an optimal schedule $\bar{S}' = (S'_1, \ldots, S'_m)$ for I' with $O(\log k)$ applications of the fpt-algorithm for Makespan Scheduling.

...scheduling big jobs

Schedule big jobs $1, \ldots, n'$ (processing times $p_i \geq L/k$) according to the oprimal schedule $\bar{S}'$ for the new instance I'

Properties of p'_i

$+$

for an opt. solution $\bar{S}$ for the whole instance I:

$S_1 \cap [n'], \ldots, S_m \cap [n']$ is a feasible solution for I' $\quad \Longrightarrow$

$$\max_{1 \leq j \leq m} \sum_{i \in S'_j} p_i \leq \cdots \leq \cdots\cdots\cdots\cdots$$
$$\leq (1 + 1/k) \cdot \max_{1 \leq j \leq m} \sum_{i \in S_j \cap [n']} p_i$$

...scheduling big jobs

Schedule big jobs $1, \ldots, n'$ (processing times $p_i \geq L/k$) according to the oprimal schedule $\bar{S}'$ for the new instance I'

Properties of p'_i

$+$

for an opt. solution $\bar{S}$ for the whole instance I:

$S_1 \cap [n'], \ldots, S_m \cap [n']$ is a feasible solution for I' $\implies$

$$\max_{1 \leq j \leq m} \sum_{i \in S'_j} p_i \leq \cdots \leq \cdots\cdots\cdots\cdots$$

$$\leq (1 + 1/k) \cdot \max_{1 \leq j \leq m} \sum_{i \in S_j \cap [n']} p_i$$

$$= (1 + 1/k) \cdot OPT(I)$$

- $n' = n$: DONE

...scheduling big jobs and small jobs

Schedule big jobs $1, \ldots, n'$ (processing times $p_i \geq L/k$) according to the oprimal schedule $\bar{S}'$ for the new instance I'

Properties of p'_i

$+$

for an opt. solution $\bar{S}$ for the whole instance I:

$S_1 \cap [n'], \ldots, S_m \cap [n']$ is a feasible solution for I' $\implies$

$$\max_{1 \leq j \leq m} \sum_{i \in S'_j} p_i \leq \cdots \leq \cdots\cdots\cdots\cdots$$
$$\leq (1 + 1/k) \cdot \max_{1 \leq j \leq m} \sum_{i \in S_j \cap [n']} p_i$$
$$= (1 + 1/k) \cdot OPT(I)$$

- $n' = n$: DONE
- $n' < n$: schedule the small ($p_i < L/k$) jobs $n' + 1, \ldots, n$ greedily, obtain a schedule $\bar{S}^*$

$$\max_{1 \leq j \leq m} \sum_{i \in S^*_j} p_i \leq (1 + 1/k) \cdot OPT(I)$$