Scheduling with Precedence Constraints of Low Fractional Dimension

Nikos Mutsanas (nikos@idsia.ch)

joint work with

Christoph Ambühl Monaldo Mastrolilli Ola Svensson

Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA)
Manno, Switzerland

Approximation Algorithms

Many practical problems are unlikely to be solvable *efficiently*.
The necessity to solve them *somehow* leads to...

Approximation Algorithm (for Minimization Problem)

A ρ-approximation algorithm ($\rho \geq 1$) satisfies:

- it runs in polynomial time
- it produces a feasible solution A
- $\text{val}(A) \leq \rho \cdot OPT$.

Approximation Algorithms

Many practical problems are unlikely to be solvable *efficiently*.
The necessity to solve them *somehow* leads to...

Approximation Algorithm (for Minimization Problem)

A ρ-approximation algorithm ($\rho \geq 1$) satisfies:

- it runs in polynomial time
- it produces a feasible solution A
- $\text{val}(A) \leq \rho \cdot OPT$.

E.g: 2-approximation for Vertex Cover: covers the graph with at most twice as many vertices as necessary.

Approximation Algorithms

Many practical problems are unlikely to be solvable *efficiently*.
The necessity to solve them *somehow* leads to...

Approximation Algorithm (for Minimization Problem)

A ρ-approximation algorithm ($\rho \geq 1$) satisfies:

- it runs in polynomial time
- it produces a feasible solution A
- $\text{val}(A) \leq \rho \cdot OPT$.

Research in Approximation is two-fold

Approximation Algorithms

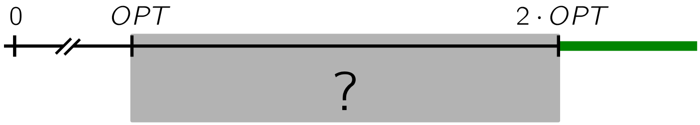
Many practical problems are unlikely to be solvable *efficiently*.
The necessity to solve them *somehow* leads to...

Approximation Algorithm (for Minimization Problem)

A ρ-approximation algorithm ($\rho \geq 1$) satisfies:

- it runs in polynomial time
- it produces a feasible solution A
- $\text{val}(A) \leq \rho \cdot OPT$.

Research in Approximation is two-fold



Approximation Algorithms

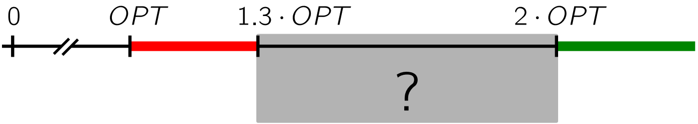
Many practical problems are unlikely to be solvable *efficiently*.
The necessity to solve them *somehow* leads to...

Approximation Algorithm (for Minimization Problem)

A ρ-approximation algorithm ($\rho \geq 1$) satisfies:

- it runs in polynomial time
- it produces a feasible solution A
- $\text{val}(A) \leq \rho \cdot OPT$.

Research in Approximation is two-fold



Approximation Algorithms

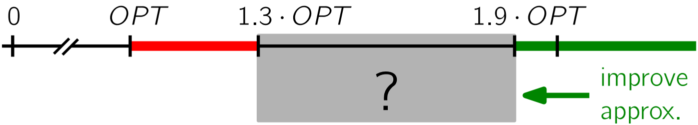
Many practical problems are unlikely to be solvable *efficiently*.
The necessity to solve them *somehow* leads to...

Approximation Algorithm (for Minimization Problem)

A ρ-approximation algorithm ($\rho \geq 1$) satisfies:

- it runs in polynomial time
- it produces a feasible solution A
- $\text{val}(A) \leq \rho \cdot OPT$.

Research in Approximation is two-fold



Approximation Algorithms

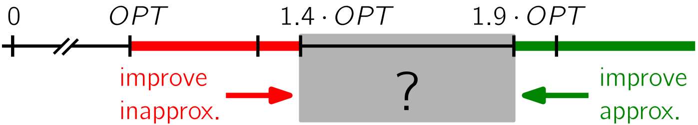
Many practical problems are unlikely to be solvable *efficiently*.
The necessity to solve them *somehow* leads to...

Approximation Algorithm (for Minimization Problem)

A ρ-approximation algorithm ($\rho \geq 1$) satisfies:

- it runs in polynomial time
- it produces a feasible solution A
- $\text{val}(A) \leq \rho \cdot OPT$.

Research in Approximation is two-fold



Approximation Algorithms

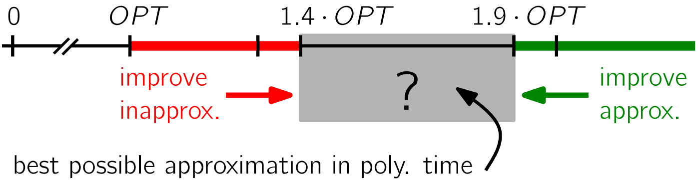
Many practical problems are unlikely to be solvable *efficiently*.
The necessity to solve them *somehow* leads to...

Approximation Algorithm (for Minimization Problem)

A ρ-approximation algorithm ($\rho \geq 1$) satisfies:

- it runs in polynomial time
- it produces a feasible solution A
- $\text{val}(A) \leq \rho \cdot OPT$.

Research in Approximation is two-fold

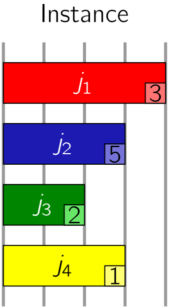


A simple scheduling problem ($1||\sum_j w_j C_j$)

Given: 1 machine and n jobs, each with p_i, w_i.
Find: An ordering of the jobs, such that $\sum_i w_i C_i$ is minimized.

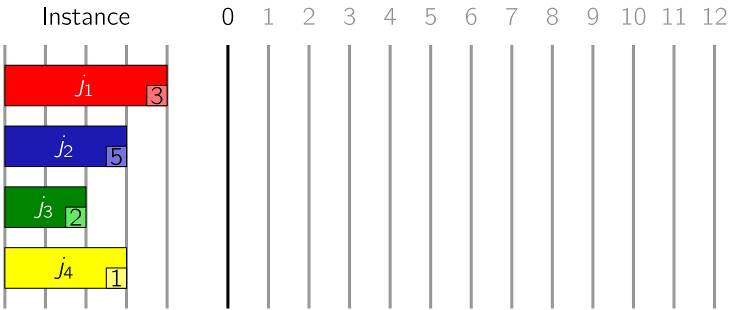
A simple scheduling problem ($1||\sum_j w_j C_j$)

Given: 1 machine and n jobs, each with p_i, w_i.
Find: An ordering of the jobs, such that $\sum_i w_i C_i$ is minimized.



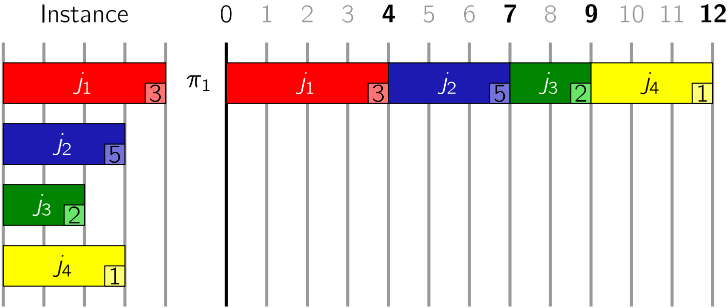
A simple scheduling problem ($1||\sum_j w_j C_j$)

Given: 1 machine and n jobs, each with p_i, w_i.
Find: An ordering of the jobs, such that $\sum_i w_i C_i$ is minimized.



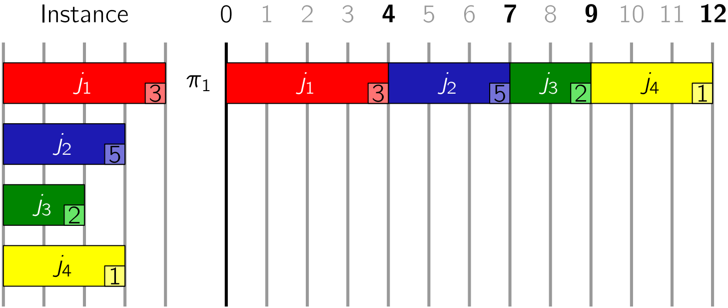
A simple scheduling problem ($1||\sum_j w_j C_j$)

Given: 1 machine and n jobs, each with p_i, w_i.
Find: An ordering of the jobs, such that $\sum_i w_i C_i$ is minimized.



A simple scheduling problem ($1||\sum_j w_j C_j$)

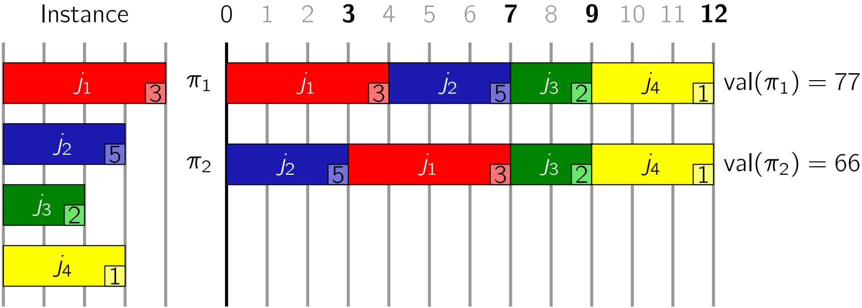
Given: 1 machine and n jobs, each with p_i, w_i.
Find: An ordering of the jobs, such that $\sum_i w_i C_i$ is minimized.



$$\text{val}(\pi_1) = 3 \cdot 4 + 5 \cdot 7 + 2 \cdot 9 + 1 \cdot 12 = 77$$

A simple scheduling problem ($1||\sum_j w_j C_j$)

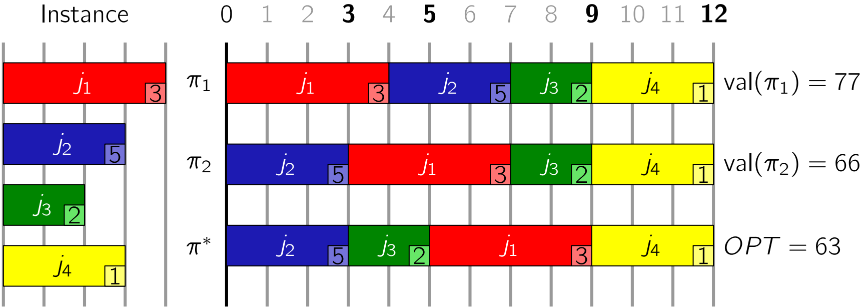
Given: 1 machine and n jobs, each with p_i, w_i.
Find: An ordering of the jobs, such that $\sum_i w_i C_i$ is minimized.



$$\text{val}(\pi_2) = 5 \cdot 3 + 3 \cdot 7 + 2 \cdot 9 + 1 \cdot 12 = 66$$

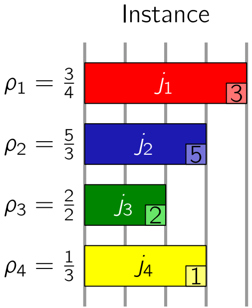
A simple scheduling problem ($1||\sum_j w_j C_j$)

Given: 1 machine and n jobs, each with p_i, w_i.
Find: An ordering of the jobs, such that $\sum_i w_i C_i$ is minimized.



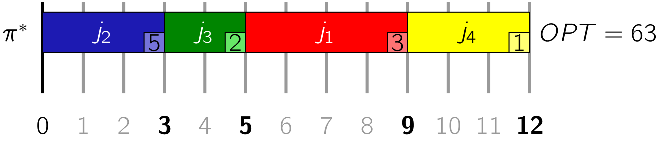
A simple scheduling problem ($1||\sum_j w_j C_j$)

Given: 1 machine and n jobs, each with p_i, w_i.
Find: An ordering of the jobs, such that $\sum_i w_i C_i$ is minimized.



Theorem [SMITH'56]

Ordering non-increasingly according to $\rho := w/p$ is optimal.



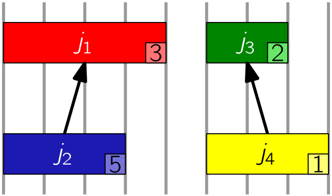
Definition $1|\text{prec}|\sum_j w_j C_j$

Given: 1 machine and n jobs, each with p_i, w_i.
Find: An ordering of the jobs, such that $\sum_i w_i C_i$ is minimized.

Definition $1|\text{prec}|\sum_j w_j C_j$

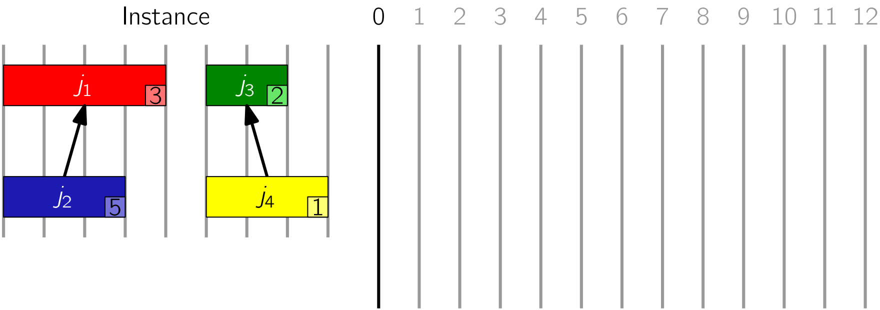
Given: 1 machine and n jobs, each with p_i, w_i.
Find: An ordering of the jobs, such that $\sum_i w_i C_i$ is minimized.

Instance



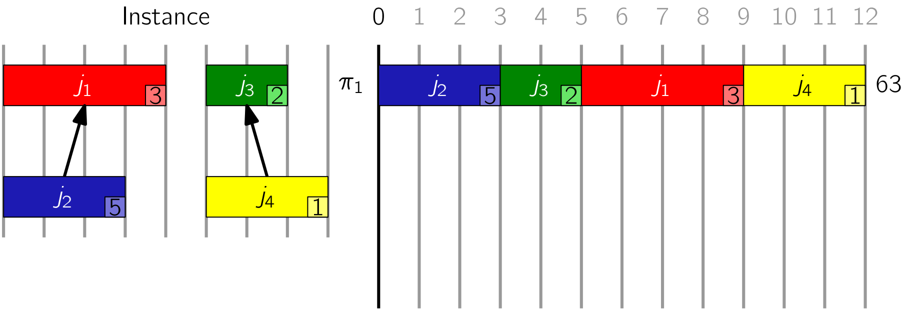
Definition $1|\text{prec}|\sum_j w_j C_j$

Given: 1 machine and n jobs, each with p_i, w_i.
Find: An ordering of the jobs, such that $\sum_i w_i C_i$ is minimized.



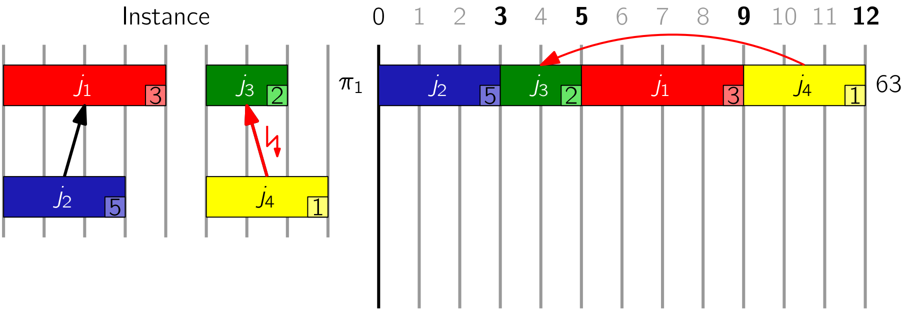
Definition $1|\text{prec}|\sum_j w_j C_j$

Given: 1 machine and n jobs, each with p_i, w_i.
Find: An ordering of the jobs, such that $\sum_i w_i C_i$ is minimized.



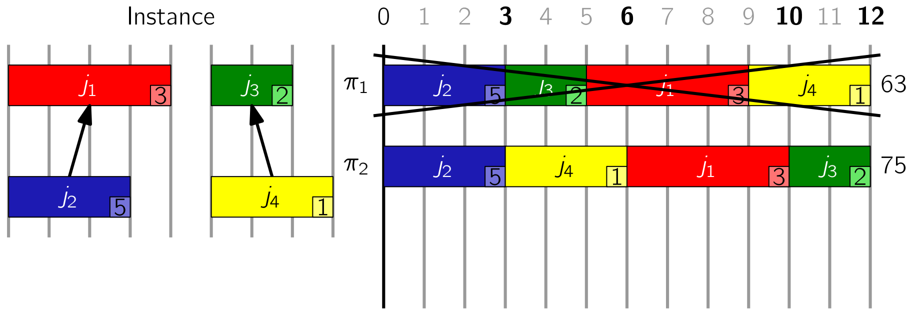
Definition $1|\text{prec}|\sum_j w_j C_j$

Given: 1 machine and n jobs, each with p_i, w_i.
Find: An ordering of the jobs, such that $\sum_i w_i C_i$ is minimized.



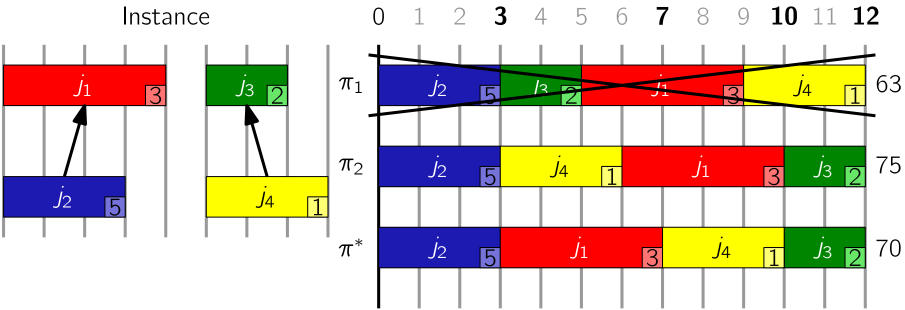
Definition $1|prec|\sum_j w_j C_j$

Given: 1 machine and n jobs, each with p_i, w_i.
Find: An ordering of the jobs, such that $\sum_i w_i C_i$ is minimized.



Definition $1|\text{prec}|\sum_j w_j C_j$

Given: 1 machine and n jobs, each with p_i, w_i.
Find: An ordering of the jobs, such that $\sum_i w_i C_i$ is minimized.



History & Literature

1. Extensively studied:
   - The general version is strongly NP-hard
     [Lawler'78], [Lenstra & Rinnooy-Kan'78]
   - Several 2-approximation algorithms
     [Schulz'96], [Hall,Schulz, Shmoys & Wein'97], [Chudak & Hochbaum'97], [Chekuri & Motwani'99], [Margot, Queyranne & Wang'03], [Pisaruk'03]
   - Better than 2-approximation for special precedence constraints
     [Woeginger'03], [Kolliopoulos & Steiner'02], [Correa & Schulz'04], [Ambühl, Mastrolilli & Svensson'06], [Ambühl, Mastrolilli, Mutsanas & Svensson'07]
   - Special cases of precedence constraints are solvable in poly-time
     [Lawler'78], [Möhring'89], [Goemans & Williamson'00], [Ambühl & Mastrolilli'06]
   - It is a vertex cover problem [Correa & Schulz'04], [Ambühl & Mastrolilli'06]
2. Inapproximability results
   - No PTAS. Variable part as hard as vertex cover
     [Ambühl, Mastrolilli & Svensson'07]
   - Closing the approximability gap is a prominent open problem
     Open Problem 9 in [Schuurman & Woeginger'99]
   - Not better than 2, assuming variant of UGC
     [Bansal & Khot '09]

History & Literature

1. Extensively studied:
   - The general version is strongly NP-hard
     [Lawler'78], [Lenstra & Rinnooy-Kan'78]
   - Several 2-approximation algorithms
     [Schulz'96], [Hall,Schulz, Shmoys & Wein'97], [Chudak & Hochbaum'97], [Chekuri & Motwani'99], [Margot, Queyranne & Wang'03], [Pisaruk'03]
   - Better than 2-approximation for special precedence constraints
     [Woeginger'03], [Kolliopoulos & Steiner'02], [Correa & Schulz'04], [Ambühl, Mastrolilli & Svensson'06], [Ambühl, Mastrolilli, Mutsanas & Svensson'07]
   - Special cases of precedence constraints are solvable in poly-time
     [Lawler'78], [Möhring'89], [Goemans & Williamson'00], [Ambühl & Mastrolilli'06]
   - It is a vertex cover problem [Correa & Schulz'04], [Ambühl & Mastrolilli'06]
2. Inapproximability results
   - No PTAS. Variable part as hard as vertex cover
     [Ambühl, Mastrolilli & Svensson'07]
   - Closing the approximability gap is a prominent open problem
     Open Problem 9 in [Schuurman & Woeginger'99]
   - Not better than 2, assuming variant of UGC
     [Bansal & Khot '09]

IDSIA

History & Literature

1. Extensively studied:
   - The general version is strongly NP-hard
     [Lawler'78], [Lenstra & Rinnooy-Kan'78]
   - Several 2-approximation algorithms
     [Schulz'96], [Hall,Schulz, Shmoys & Wein'97], [Chudak & Hochbaum'97], [Chekuri & Motwani'99], [Margot, Queyranne & Wang'03], [Pisaruk'03]
   - Better than 2-approximation for special precedence constraints
     [Woeginger'03], [Kolliopoulos & Steiner'02], [Correa & Schulz'04], [Ambühl, Mastrolilli & Svensson'06], [Ambühl, Mastrolilli, Mutsanas & Svensson'07]
   - Special cases of precedence constraints are solvable in poly-time
     [Lawler'78], [Möhring'89], [Goemans & Williamson'00], [Ambühl & Mastrolilli'06]
   - It is a vertex cover problem [Correa & Schulz'04], [Ambühl & Mastrolilli'06]
2. Inapproximability results
   - No PTAS. Variable part as hard as vertex cover
     [Ambühl, Mastrolilli & Svensson'07]
   - Closing the approximability gap is a prominent open problem
     Open Problem 9 in [Schuurman & Woeginger'99]
   - Not better than 2, assuming variant of UGC
     [Bansal & Khot '09]

Motivation & Results

Hard problem

We don't know of a better than 2-approximation algorithm for the general case.

Motivation & Results

Hard problem

We don't know of a better than 2-approximation algorithm for the general case.

Impose restrictions on...

weights / proc. times

Severe restrictions on w and p do not affect approximability [Woeginger'03]

$\Rightarrow$

precedence constraints

Dimension theory of partial orders provides a well established characterization of precedence constraints [Trotter'92]

IDSIA

Motivation & Results

Hard problem

We don't know of a better than 2-approximation algorithm for the general case.

Impose restrictions on. . .

weights / proc. times

Severe restrictions on w and p do not affect approximability [Woeginger'03]

$\Rightarrow$

precedence constraints

Dimension theory of partial orders provides a well established characterization of precedence constraints [Trotter'92]

Motivation & Results

Hard problem

We don't know of a better than 2-approximation algorithm for the general case.

Impose restrictions on...

weights / proc. times

Severe restrictions on w and p do not affect approximability [Woeginger'03]

$\Rightarrow$

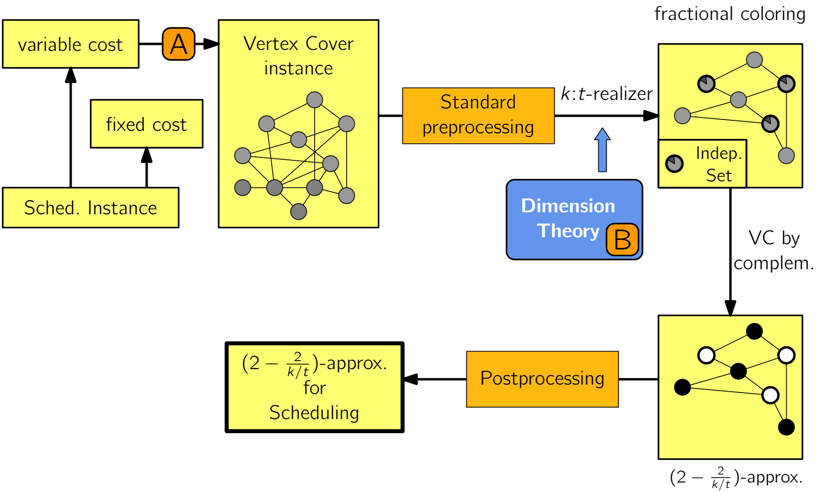
precedence constraints

Dimension theory of partial orders provides a well established characterization of precedence constraints [Trotter'92]

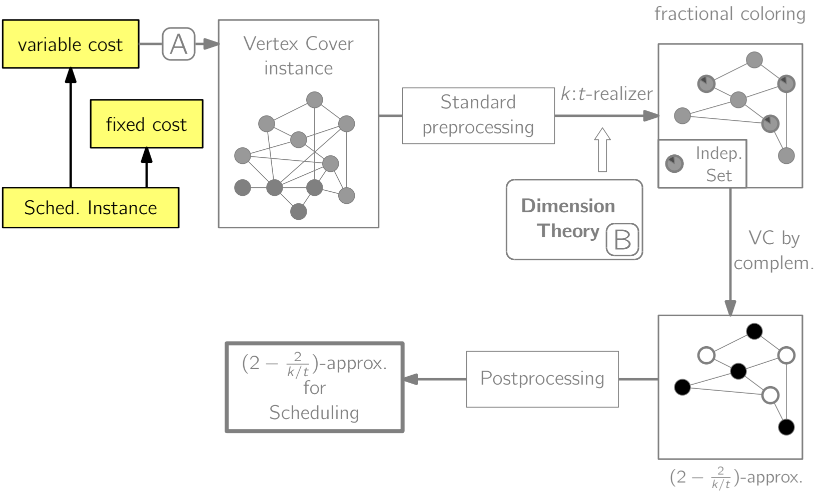
General framework

Yields a $(2 - 2/f)$-approximation algorithm whenever the fractional dimension of the poset is $\leq f$. (interval orders, bounded degree, ...)

The algorithmic framework - Overview



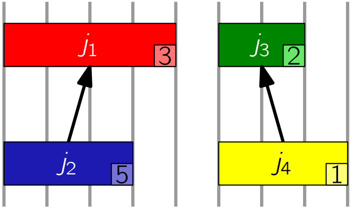
The algorithmic framework - Fixed cost



The algorithmic framework - Fixed cost

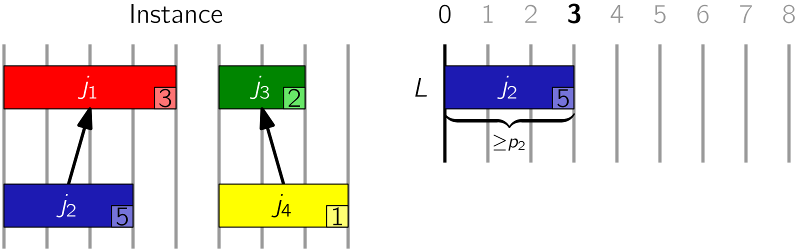
$$\text{val}(L) = \sum_j w_j C_j$$

Instance



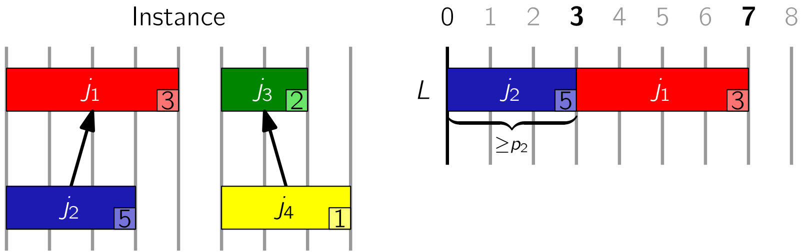
The algorithmic framework - Fixed cost

$$\text{val}(L) = \sum_j w_j C_j = \sum_j w_j p_j + \sum_{(i,j)\in L} w_j p_i$$



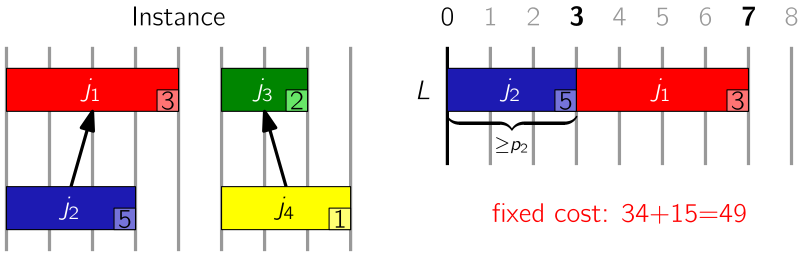
The algorithmic framework - Fixed cost

$$\text{val}(L) = \sum_j w_j C_j = \sum_j w_j p_j + \sum_{(i,j)\in L} w_j p_i = \underbrace{\overbrace{\sum_j w_j p_j}^{\text{own proc. time}} + \overbrace{\sum_{(i,j)\in P} w_j p_i}^{\text{poset}}}_{\text{fixed cost}} + \underbrace{\sum_{(i,j)\in L\setminus P} w_j p_i}_{\text{variable cost}}$$

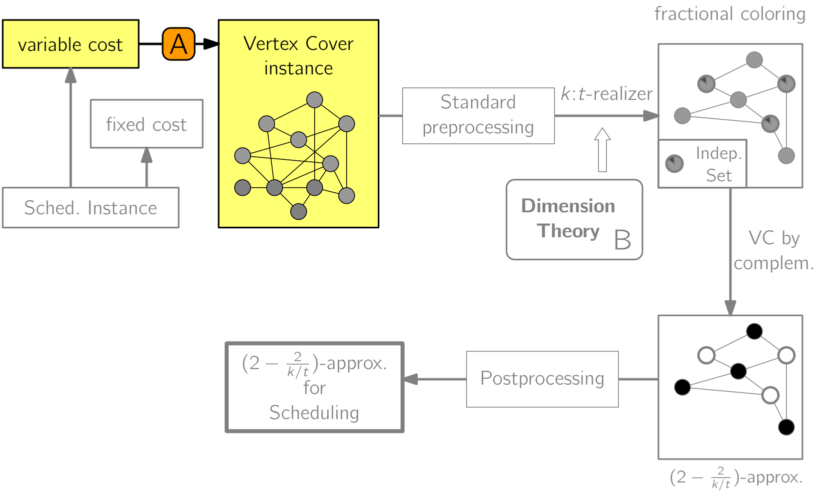


The algorithmic framework - Fixed cost

$$\text{val}(L) = \sum_j w_j C_j = \sum_j w_j p_j + \sum_{(i,j)\in L} w_j p_i = \underbrace{\overbrace{\sum_j w_j p_j}^{\text{own proc. time}} + \overbrace{\sum_{(i,j)\in P} w_j p_i}^{\text{poset}}}_{\text{fixed cost}} + \underbrace{\sum_{(i,j)\in L\setminus P} w_j p_i}_{\text{variable cost}}$$



The algorithmic framework - Sched. & Vertex Cover



The algorithmic framework - Sched. & Vertex Cover

Theorem

Problem $1|prec|\sum_j w_j C_j$ *is a special case of* MINIMUM WEIGHTED VERTEX COVER.

- Obtained by studying several IP-formulations of $1|\text{prec}|\sum_j w_j C_j$.

The algorithmic framework - Sched. & Vertex Cover

Theorem

Problem $1|prec|\sum_j w_j C_j$ *is a special case of* MINIMUM WEIGHTED VERTEX COVER.

- Obtained by studying several IP-formulations of $1|\text{prec}|\sum_j w_j C_j$.

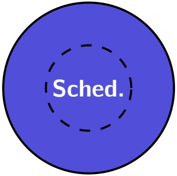
[Potts'80]

The algorithmic framework - Sched. & Vertex Cover

Theorem

Problem $1|prec|\sum_j w_j C_j$ *is a special case of* MINIMUM WEIGHTED VERTEX COVER.

- Obtained by studying several IP-formulations of $1|\text{prec}|\sum_j w_j C_j$.



[Potts'80]

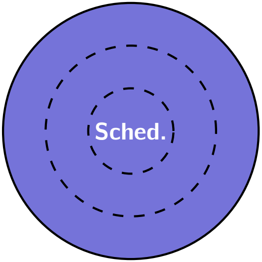
[Chudak & Hochbaum'99]

The algorithmic framework - Sched. & Vertex Cover

Theorem

Problem $1|prec|\sum_j w_jC_j$ *is a special case of* MINIMUM WEIGHTED VERTEX COVER.

- Obtained by studying several IP-formulations of $1|\text{prec}|\sum_j w_jC_j$.



[Potts'80]

[Chudak & Hochbaum'99]

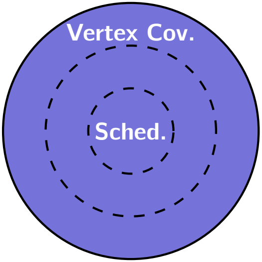
[Correa & Schulz'04]

The algorithmic framework - Sched. & Vertex Cover

Theorem

Problem $1|prec|\sum_j w_j C_j$ *is a special case of* MINIMUM WEIGHTED VERTEX COVER*.*

- Obtained by studying several IP-formulations of $1|\text{prec}|\sum_j w_j C_j$.



[Potts'80]

[Chudak & Hochbaum'99]

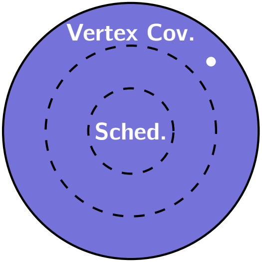
[Correa & Schulz'04]

The algorithmic framework - Sched. & Vertex Cover

Theorem

Problem $1|prec|\sum_j w_j C_j$ *is a special case of* MINIMUM WEIGHTED VERTEX COVER*.*

- Obtained by studying several IP-formulations of $1|\text{prec}|\sum_j w_j C_j$.



[Potts'80]

[Chudak & Hochbaum'99]

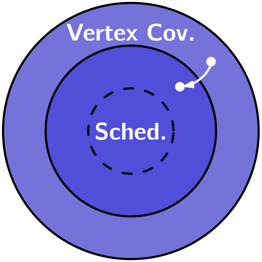
[Correa & Schulz'04]

The algorithmic framework - Sched. & Vertex Cover

Theorem

Problem $1|prec|\sum_j w_j C_j$ *is a special case of* MINIMUM WEIGHTED VERTEX COVER.

- Obtained by studying several IP-formulations of $1|\text{prec}|\sum_j w_j C_j$.



[Potts'80]

[Chudak & Hochbaum'99]

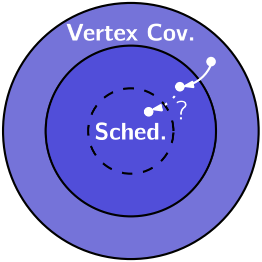
[Correa & Schulz'04]

The algorithmic framework - Sched. & Vertex Cover

Theorem

Problem $1|prec|\sum_j w_jC_j$ *is a special case of* MINIMUM WEIGHTED VERTEX COVER.

- Obtained by studying several IP-formulations of $1|\text{prec}|\sum_j w_jC_j$.



[Potts'80]

[Chudak & Hochbaum'99]

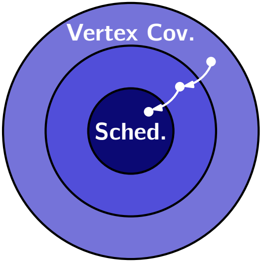
[Correa & Schulz'04]

The algorithmic framework - Sched. & Vertex Cover

Theorem

Problem $1|prec|\sum_j w_j C_j$ *is a special case of* MINIMUM WEIGHTED VERTEX COVER.

- Obtained by studying several IP-formulations of $1|\text{prec}|\sum_j w_j C_j$.



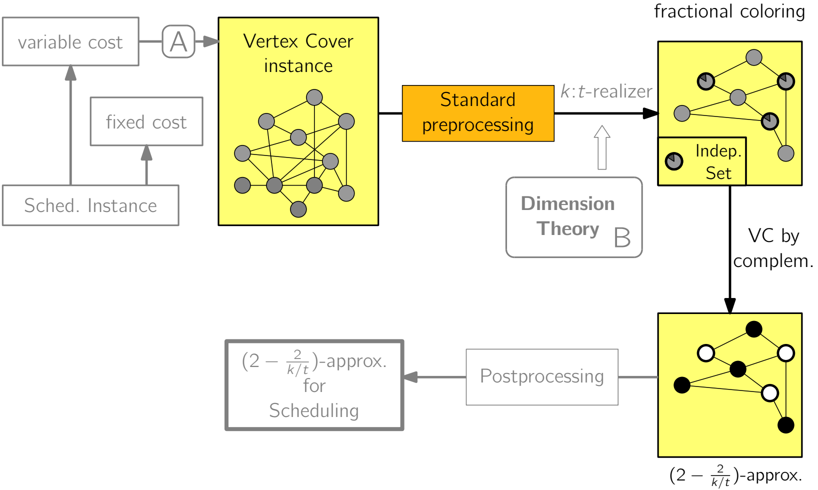
[Potts'80]

[Chudak & Hochbaum'99]

[Correa & Schulz'04]

[Ambühl & Mastrolilli'06]

The algorithmic framework - VC & coloring

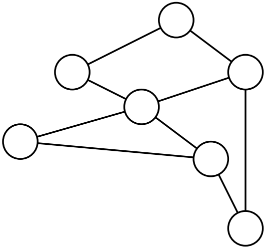


The algorithmic framework - VC & coloring

Given: Graph $G(V, E)$.
Find: Coloring of V s.t. no $e \in E$ is "monochromatic".

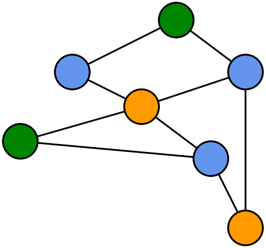
The algorithmic framework - VC & coloring

Given: Graph $G(V, E)$.
Find: Coloring of V s.t. no $e \in E$ is "monochromatic".



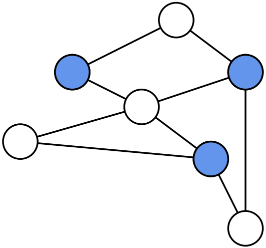
The algorithmic framework - VC & coloring

Given: Graph $G(V, E)$.
Find: Coloring of V s.t. no $e \in E$ is "monochromatic".



The algorithmic framework - VC & coloring

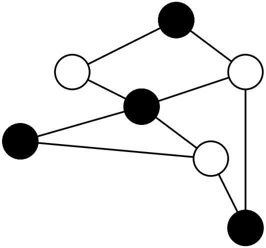
Given: Graph $G(V, E)$.
Find: Coloring of V s.t. no $e \in E$ is "monochromatic".



[HOCHBAUM'83]
The **heaviest color** weighs at least $\frac{W}{k}$ (average weight).

The algorithmic framework - VC & coloring

Given: Graph $G(V, E)$.
Find: Coloring of V s.t. no $e \in E$ is “monochromatic”.

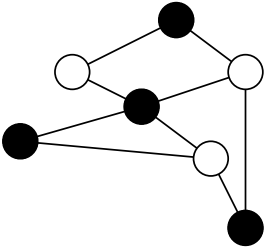


[HOCHBAUM’83]
The **heaviest color** weighs at least $\frac{W}{k}$ (average weight).

Define VC by taking the complement.

The algorithmic framework - VC & coloring

Given: Graph $G(V, E)$.
Find: Coloring of V s.t. no $e \in E$ is "monochromatic".



[HOCHBAUM'83]
The **heaviest color** weighs at least $\frac{W}{k}$ (average weight).

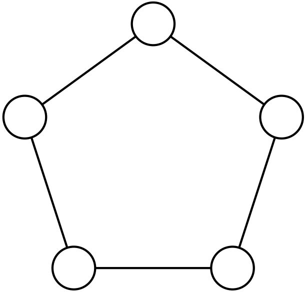
Define VC by taking the complement.

$$\frac{\text{val}(A)}{OPT} \leq \frac{W - W/k}{W/2} \leq \left(2 - \frac{2}{k}\right)$$

The algorithmic framework - VC & *fractional* coloring

The algorithmic framework - VC & *fractional* coloring

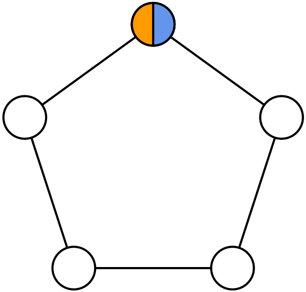
Fractional Coloring: color each vertex t times using a k-palette, s.t. k/t is minimized.



$k = 5, \quad t = 2$

The algorithmic framework - VC & *fractional* coloring

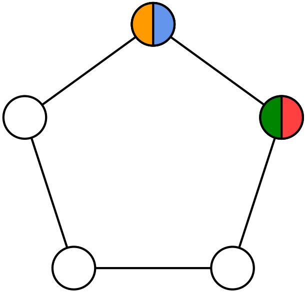
Fractional Coloring: color each vertex t times using a k-palette, s.t. k/t is minimized.



$k = 5, \quad t = 2$

The algorithmic framework - VC & *fractional* coloring

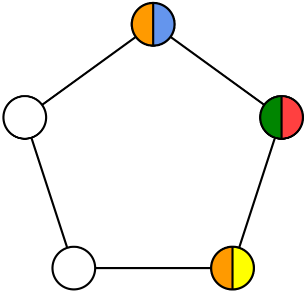
Fractional Coloring: color each vertex t times using a k-palette, s.t. k/t is minimized.



$k = 5, \quad t = 2$

The algorithmic framework - VC & *fractional* coloring

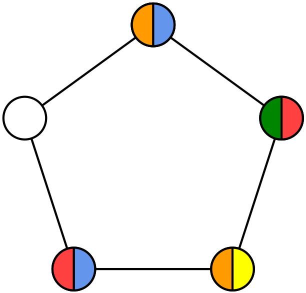
Fractional Coloring: color each vertex t times using a k-palette, s.t. k/t is minimized.



$k = 5, \quad t = 2$

The algorithmic framework - VC & *fractional* coloring

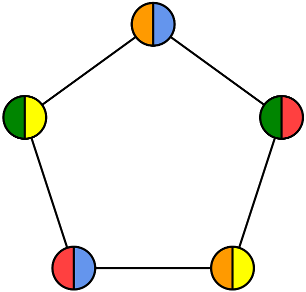
Fractional Coloring: color each vertex t times using a k-palette, s.t. k/t is minimized.



$k = 5, \quad t = 2$

The algorithmic framework - VC & *fractional* coloring

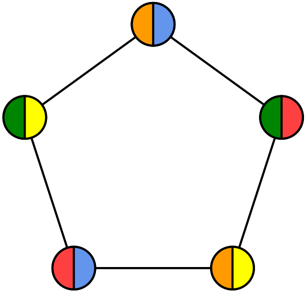
Fractional Coloring: color each vertex t times using a k-palette, s.t. k/t is minimized.



$k = 5, \quad t = 2$

The algorithmic framework - VC & *fractional* coloring

Fractional Coloring: color each vertex t times using a k-palette, s.t. k/t is minimized.



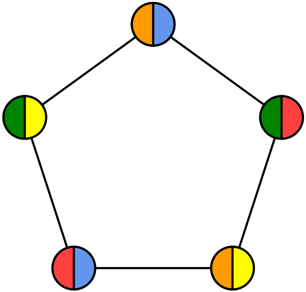
$k = 5, \quad t = 2$

[THIS APPROACH]

The **heaviest color** weighs at least $\frac{t \cdot W}{k}$ (average weight).

The algorithmic framework - VC & *fractional* coloring

Fractional Coloring: color each vertex t times using a k-palette, s.t. k/t is minimized.



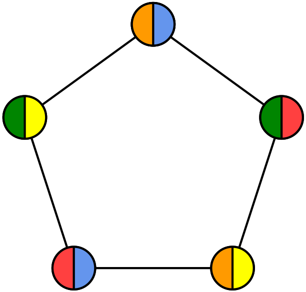
$k = 5, \quad t = 2$

[THIS APPROACH]
The **heaviest color** weighs at least $\frac{t \cdot W}{k}$ (average weight).

Define VC by taking the complement.

The algorithmic framework - VC & *fractional* coloring

Fractional Coloring: color each vertex t times using a k-palette, s.t. k/t is minimized.



$k = 5, \quad t = 2$

[THIS APPROACH]
The **heaviest color** weighs at least $\frac{t \cdot W}{k}$ (average weight).

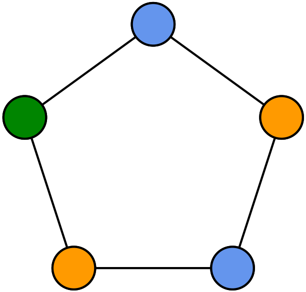
Define VC by taking the complement.

$$\frac{\text{val}(A)}{OPT} \leq \frac{W - t \cdot W/k}{W/2} \leq \left(2 - \frac{2}{k/t}\right)$$

Here: $\frac{6}{5}$

The algorithmic framework - VC & *fractional* coloring

Fractional Coloring: color each vertex t times using a k-palette, s.t. k/t is minimized.



[THIS APPROACH]
The **heaviest color** weighs at least $\frac{t \cdot W}{k}$ (average weight).

Define VC by taking the complement.

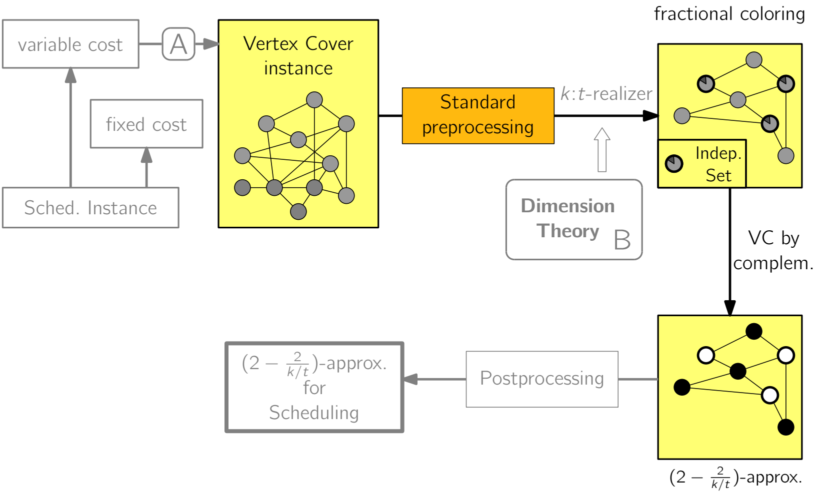
$$\frac{\mathrm{val}(A)}{OPT} \leq \frac{W - t \cdot W/k}{W/2} \leq \left(2 - \frac{2}{k/t}\right)$$

Here: $\frac{6}{5}$

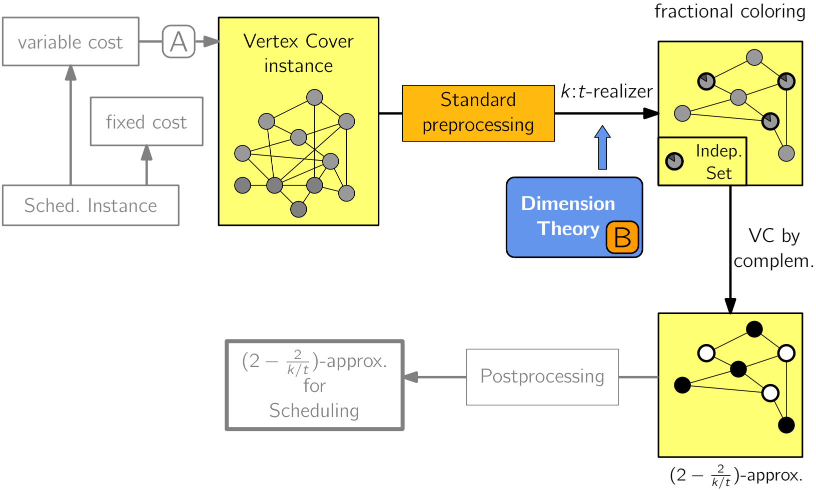
Using coloring [HOCHBAUM'83]:

$$\left(2 - \frac{2}{3}\right) = \frac{4}{3}$$

The algorithmic framework - VC & *fractional* coloring

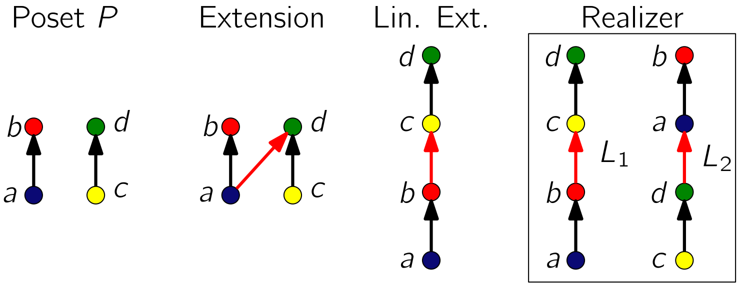


The algorithmic framework - VC & *fractional* coloring



IDSIA

The algorithmic framework - Dimension Theory



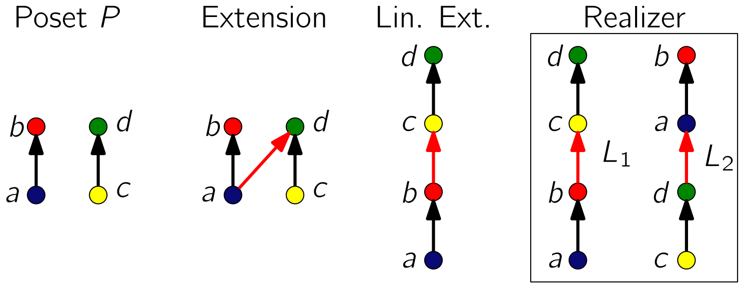
Definition

A *t*-realizer of a poset **P** is a set of *t* linear extensions of **P** s.t. any (ordered) incomparable pair is reversed in at least 1 linear extension .

Definition [DUSHNIK & MILLER, 1941]

The dimension of a poset **P** is the smallest *t* such that there exists a *t*-realizer of *P*.

The algorithmic framework - Dimension Theory



Definition

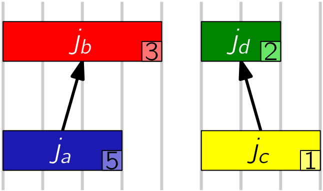
A *k*:*t*-realizer of a poset **P** is a set of *t* linear extensions of **P** s.t. any (ordered) incomparable pair is reversed in at least *k* linear extensions.

Definition [Brightwell & Scheinerman'92]

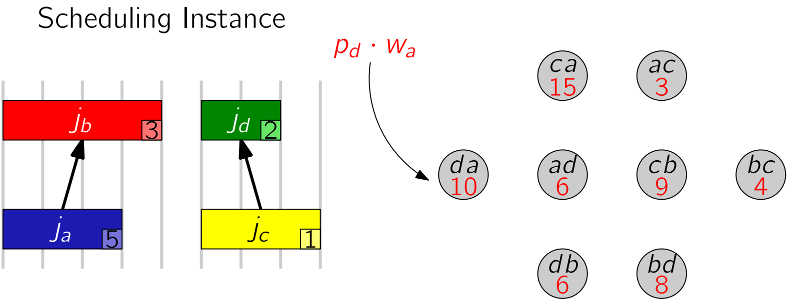
The fractional dimension of a poset **P** is the smallest t/k such that there exists a *k*:*t*-realizer of *P*.

The algorithmic framework - Graph Structure

Scheduling Instance

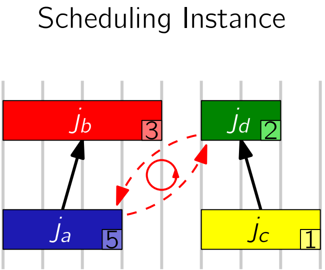


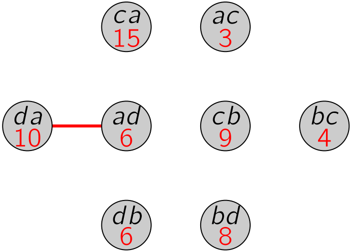
The algorithmic framework - Graph Structure



- A vertex for each (ordered) pair of incomparable jobs.
- Intuitively: If $(a, d) \in VC$ then job a is scheduled before job d.

The algorithmic framework - Graph Structure

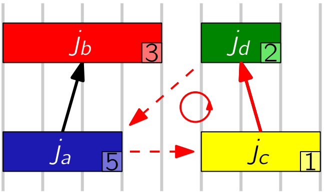


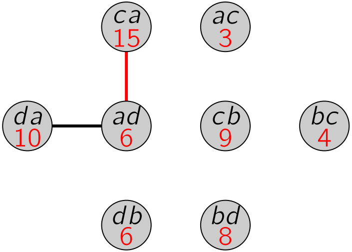


- Edge type (1): Either schedule a before d or d before a.

The algorithmic framework - Graph Structure

Scheduling Instance

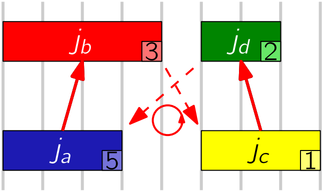


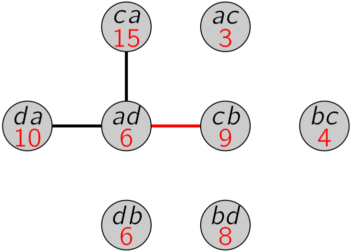


- Edge type (2): Schedule *c* before *a* or *a* before *d* to avoid cycles of this type.

The algorithmic framework - Graph Structure

Scheduling Instance

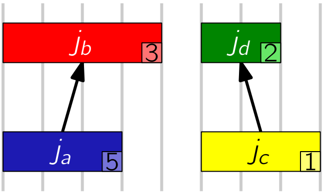


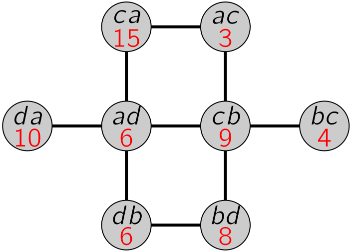


- Edge type (3): Schedule c before b or a before d to avoid cycles of this type.

The algorithmic framework - Graph Structure

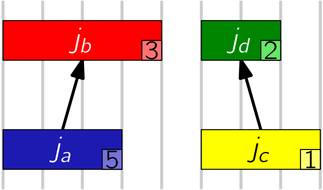
Scheduling Instance



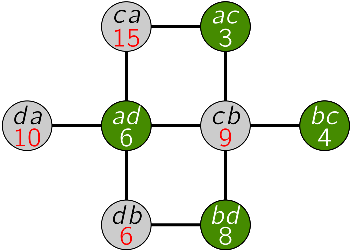


The algorithmic framework - Graph Structure

Scheduling Instance

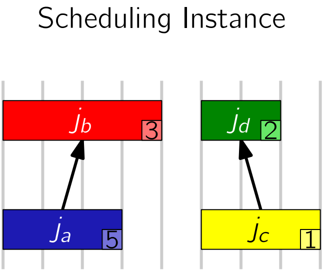


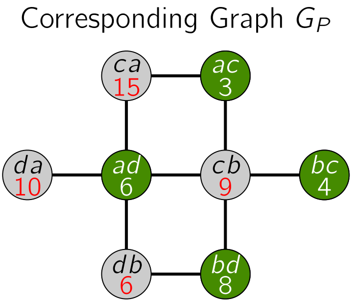
Corresponding Graph G_P



- The green nodes represent an optimal vertex cover with value $3 + 6 + 8 + 4 = 21$ (variable part).

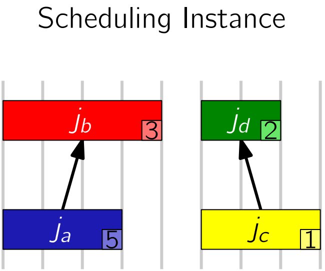
The algorithmic framework - Graph Structure

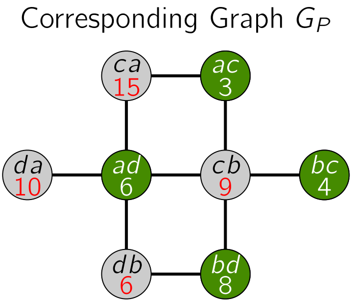




- Observe that any linear extension L defines a vertex cover of the graph. [$L = (a, b, c, d)$ defines **this vertex cover**.]

The algorithmic framework - Graph Structure





- Observe that any linear extension L defines a vertex cover of the graph. [$L = (a, b, c, d)$ defines **this vertex cover**.]
- Pairs that are reversed in L form an independent set in G_P.

Posets & (Hyper)graph of incomparable pairs

In Dimension theory the following Hypergraph $H_{\mathbf{P}}$ is well-known:

Vertices: (Ordered) Incomparable pairs

Hyperedges: (Minimal) Subsets of incomparable pairs that no linear extension can reverse simultaneously.

Dimension & Coloring

$\chi(H_{\mathbf{P}}) = \dim(\mathbf{P})$ [FELSNER & TROTTER'00]

$\chi_f(H_{\mathbf{P}}) = \mathrm{fdim}(\mathbf{P})$ [BRIGHTWELL & SCHEINERMANN'92]

Posets & (Hyper)graph of incomparable pairs

In Dimension theory the following Hypergraph $H_{\mathbf{P}}$ is well-known:

Vertices: (Ordered) Incomparable pairs

Hyperedges: (Minimal) Subsets of incomparable pairs that no linear extension can reverse simultaneously.

Dimension & Coloring

$\chi(H_{\mathbf{P}}) = \dim(\mathbf{P})$ [Felsner & Trotter'00]

$\chi_f(H_{\mathbf{P}}) = \mathrm{fdim}(\mathbf{P})$ [Brightwell & Scheinermann'92]

[Ambühl et al.'06] observed that the underlying graph (hyperedges of cardinality two) and the graph of [Correa & Schulz'04] **coincide**.

IDSIA

The Approximability of the Fractional Dimension

Corrolary

If we have a "small" realizer, we have a "good" coloring of the Hypergraph $\Rightarrow$ good approximation for $1|\text{prec}|\sum_j w_j C_j$.

Theorem [Jain & Hegde'06]

It is hard to approximate the (fractional) dimension of a poset with n elements within a factor $n^{0.5-\varepsilon}$ for any $\varepsilon > 0$.

IDSIA

The Approximability of the Fractional Dimension

Corrolary

If we have a "small" realizer, we have a "good" coloring of the Hypergraph $\Rightarrow$ good approximation for $1|\text{prec}|\sum_j w_j C_j$.

Theorem [Jain & Hegde'06]

It is hard to approximate the (fractional) dimension of a poset with n elements within a factor $n^{0.5-\varepsilon}$ for any $\varepsilon > 0$.

The Approximability of the Fractional Dimension

Corrolary

If we have a “small” realizer, we have a “good” coloring of the Hypergraph $\Rightarrow$ good approximation for $1|\text{prec}|\sum_j w_j C_j$.

Theorem [Jain & Hegde'06]

It is hard to approximate the (fractional) dimension of a poset with n elements within a factor $n^{0.5-\varepsilon}$ for any $\varepsilon > 0$.

However for several interesting posets we can do better...

Scheduling & Coloring - Applications

Using coloring:

Prec. Constr.	Other approaches	This approach
2-dimensional	$3/2$ [Correa & Schulz'04]	1
semi-orders	≈ 1.618 [Woeginger'03]	$4/3$
convex bipartite	≈ 1.618 [Woeginger'03]	$4/3$
interval-orders	≈ 1.618 [Woeginger'03]	2
interval dimension 2	2	2
Bounded degree d	2	2

IDSIA

Scheduling & Coloring - Applications

Using coloring:

Prec. Constr.	Other approaches	This approach
2-dimensional	$3/2$ [Correa & Schulz'04]	1
semi-orders	≈ 1.618 [Woeginger'03]	$4/3$
convex bipartite	≈ 1.618 [Woeginger'03]	$4/3$
interval-orders	≈ 1.618 [Woeginger'03]	2
interval dimension 2	2	2
Bounded degree d	2	2

For remaining classes, use

- fractional coloring
- randomization

Scheduling & Coloring - Applications

Using coloring:

Prec. Constr.	Other approaches	This approach
2-dimensional	$3/2$ [Correa & Schulz'04]	1
semi-orders	≈ 1.618 [Woeginger'03]	$4/3$
convex bipartite	≈ 1.618 [Woeginger'03]	$4/3$
interval-orders	≈ 1.618 [Woeginger'03]	2
interval dimension 2	2	2
Bounded degree d	2	2

For remaining classes, use

- fractional coloring
- randomization

Scheduling & Coloring - Applications

Using fractional coloring:

Prec. Constr.	Other approaches	This approach
2-dimensional	$3/2$ [Correa & Schulz'04]	1
semi-orders	≈ 1.618 [Woeginger'03]	$4/3$
convex bipartite	≈ 1.618 [Woeginger'03]	$4/3$
interval-orders	≈ 1.618 [Woeginger'03]	1.5
interval dimension 2	2	1.75
Bounded degree d	2	$2 - \frac{2}{d+1}$

For remaining classes, use

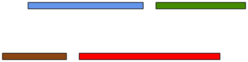
- fractional coloring
- randomization

Example - Interval Orders

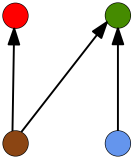
- Interval orders is a well studied class of posets [Fishburn'85]
- [Woeginger'03] showed that $1|\text{prec}|\sum_j w_j C_j$ with interval order precedence constraints has a (≈ 1.61803)-approximation.

Interval Orders

A poset is an interval order if it can be represented by intervals such that $(a, b) \in P$ iff a's interval is completely before b's.

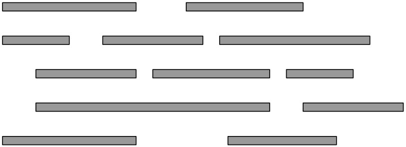


$\hat{=}$

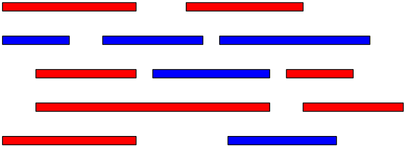


Interval orders can be recognized in $O(n^2)$

1.5-Approximation for Interval Orders



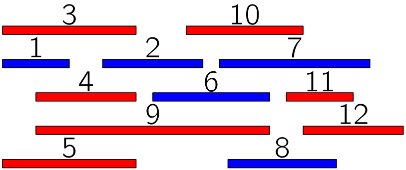
1.5-Approximation for Interval Orders



Partition the set of jobs into 2 sets (blue and red)

There are 2^n partitions

1.5-Approximation for Interval Orders



Partition the set of jobs into 2 sets (blue and red)

There are 2^n partitions

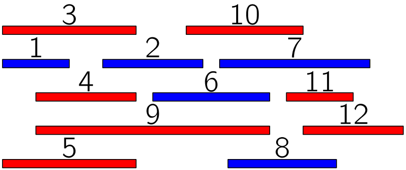
Lemma [RABINOVITCH'78]

For any (blue,red)-partition there is an L where blue jobs are scheduled before red jobs (if incomparable).

- Consider these $t = 2^n$ linear extensions
- Observe that there are $k = 2^{n-2}$ linear extensions where any inc. pair (a, b) is reversed: (a, b): **yes**, (a, b): **no**, (a, b): **maybe**, (a, b): **maybe**.
- This set of $t = 2^n$ linear extensions is a $k{:}t$-realizer ($t/k = 4$)

$$\alpha = \left(2 - \frac{2}{t/k}\right) = 2 - \frac{2}{4} = 1.5$$

1.5-Approximation for Interval Orders



Partition the set of jobs into 2 sets (blue and red)

There are 2^n partitions

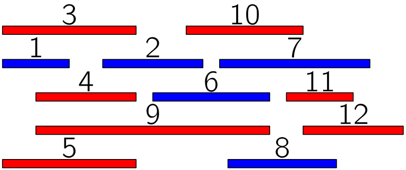
Lemma [RABINOVITCH'78]

For any (blue,red)-partition there is an L where blue jobs are scheduled before red jobs (if incomparable).

- Consider these $t = 2^n$ linear extensions
- Observe that there are $k = 2^{n-2}$ linear extensions where any inc. pair (a, b) is reversed: (a, b): **yes**, (a, b): **no**, (a, b): **maybe**, (a, b): **maybe**.
- This set of $t = 2^n$ linear extensions is a k:t-realizer $(t/k = 4)$

$$\alpha = \left(2 - \frac{2}{t/k}\right) = 2 - \frac{2}{4} = 1.5$$

1.5-Approximation for Interval Orders



Partition the set of jobs into 2 sets (blue and red)

There are 2^n partitions

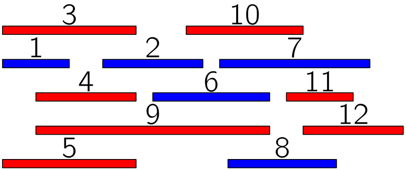
Lemma [Rabinovitch'78]

For any (blue,red)-partition there is an L where blue jobs are scheduled before red jobs (if incomparable).

- Consider these $t = 2^n$ linear extensions
- Observe that there are $k = 2^{n-2}$ linear extensions where any inc. pair (a, b) is reversed: (a, b): **yes**, (a, b): **no**, (a, b): **maybe**, (a, b): **maybe**.
- This set of $t = 2^n$ linear extensions is a k:t-realizer ($t/k = 4$)

$$\alpha = \left(2 - \frac{2}{t/k}\right) = 2 - \frac{2}{4} = 1.5$$

1.5-Approximation for Interval Orders



Partition the set of jobs into 2 sets (blue and red)

There are 2^n partitions

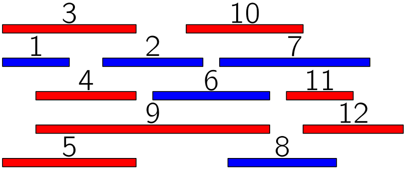
Lemma [RABINOVITCH'78]

For any (blue,red)-partition there is an L where blue jobs are scheduled before red jobs (if incomparable).

- Consider these $t = 2^n$ linear extensions
- Observe that there are $k = 2^{n-2}$ linear extensions where any inc. pair (a, b) is reversed: (a, b): **yes**, (a, b): **no**, (a, b): **maybe**, (a, b): **maybe**.
- This set of $t = 2^n$ linear extensions is a k:t-realizer ($t/k = 4$)

$$\alpha = \left(2 - \frac{2}{t/k}\right) = 2 - \frac{2}{4} = 1.5$$

1.5-Approximation for Interval Orders



Partition the set of jobs into 2 sets (blue and red)

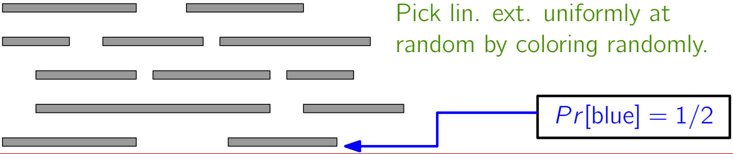
There are 2^n partitions

Problem

k:t-Realizer is of exponential size (2^n).

1.5-Approximation for Interval Orders

Pick lin. ext. uniformly at random by coloring randomly.



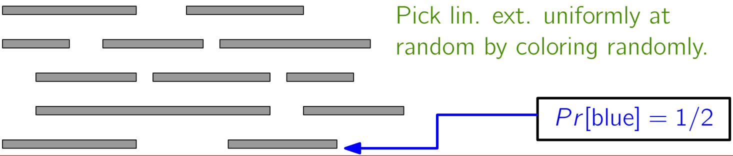
Problem

k:t-Realizer is of exponential size (2^n).

Solution

Randomization: we only need to **sample** a good extension efficiently.

1.5-Approximation for Interval Orders



Problem

k:t-Realizer is of exponential size (2^n).

Solution

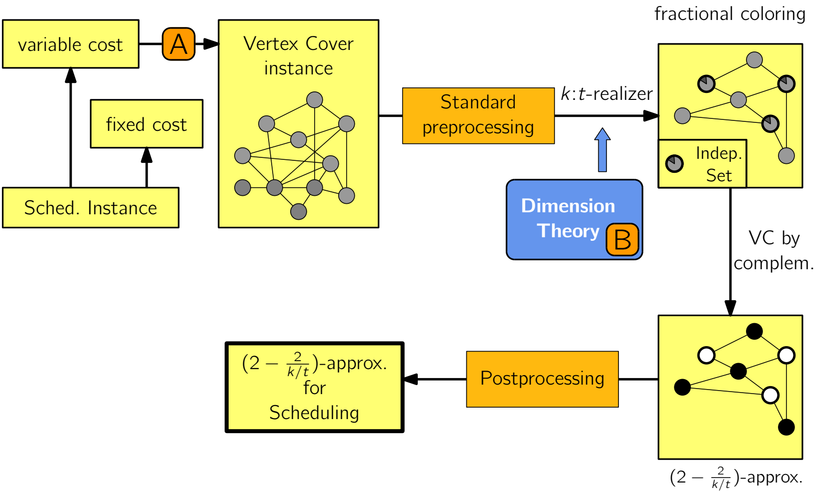
Randomization: we only need to **sample** a good extension efficiently.

Note: Randomization is merely a "detour"

Method of Conditional Probabilities

$\Rightarrow$ deterministic 1.5-approximation algorithm.

The algorithmic Framework - Applications



The algorithmic Framework - Applications

Prec. Constr.	Other approaches	This approach
2-dimensional	$3/2$ [Correa & Schulz'04]	1
semi-orders	≈ 1.618 [Woeginger'03]	$4/3$
convex bipartite	≈ 1.618 [Woeginger'03]	$4/3$
interval-orders	≈ 1.618 [Woeginger'03]	1.5
interval dimension 2	2	1.75
Bounded degree d	2	$2 - \frac{2}{d+1}$

Summary

- Scheduling problem $\rightarrow$ weighted vertex cover on $G_\mathbf{P}$ ($+$ a fixed cost)
  - Adds more structure to the problem
  - 2-approximates the variable part!
  - Suggests a unified way of constructing the currently best-known approximation ratios for all considered posets.
- We did not improve the approximation ratio for the general case
  - No better than 2-approximation, assuming variant of UGC

    [BANSAL & KHOT'09]
  - How good is SDP (...for special cases)?
  - A better understanding of the graph (e.g. when is it perfect?)

Summary

- Scheduling problem $\rightarrow$ weighted vertex cover on $G_{\mathbf{P}}$ ($+$ a fixed cost)
  - Adds more structure to the problem
  - 2-approximates the variable part!
  - Suggests a unified way of constructing the currently best-known approximation ratios for all considered posets.
- We did not improve the approximation ratio for the general case
  - No better than 2-approximation, assuming variant of UGC

    [BANSAL & KHOT'09]
  - How good is SDP (...for special cases)?
  - A better understanding of the graph (e.g. when is it perfect?)

IDSIA

Outlook

- Hypergraph of incomparable pairs has nice properties:
  - Vertex Cover $\hat{=}$ **Feedback Arc Set**
  - Independent Set $\hat{=}$ **Maximum Acyclic Subgraph**
- $1|\text{prec}|\sum_j w_j C_j$ can be seen as Feedback Arc Set with "specially structured" weights.
- "Special structure" $\Rightarrow$ all hyperedges > 2 can be ignored!
- **Hypothesis**: *Other ordering problems lie in between.*
- Work in Progress: Rank Aggregation (with triangle inequality) $\hat{=}$ ignore hyperedges > 3

IDSIA

Thank you for your attention!