Timetable Combinatorial Optimization

David Adjiashvili, Sandro Bosio, Robert Weismantel, Rico Zenklusen

IFOR, ETH Zürich, Johns Hopkins University

January 7, 2013

Motivation

Motivation

Temporal Extensions

Meaningful way to incorporate the dimension of time in Combinatorial Optimization problems.

Motivation

Temporal Extensions

Meaningful way to incorporate the dimension of time in Combinatorial Optimization problems.

Complex Constraints

Deal with complex constraints in dynamic setups.

Motivation

Temporal Extensions

Meaningful way to incorporate the dimension of time in Combinatorial Optimization problems.

Complex Constraints

Deal with complex constraints in dynamic setups.

Existing theories

Motivation

Temporal Extensions

Meaningful way to incorporate the dimension of time in Combinatorial Optimization problems.

Complex Constraints

Deal with complex constraints in dynamic setups.

Existing theories

- Scheduling

Motivation

Temporal Extensions

Meaningful way to incorporate the dimension of time in Combinatorial Optimization problems.

Complex Constraints

Deal with complex constraints in dynamic setups.

Existing theories

- Scheduling
- Flow over time

Motivation

Temporal Extensions

Meaningful way to incorporate the dimension of time in Combinatorial Optimization problems.

Complex Constraints

Deal with complex constraints in dynamic setups.

Existing theories

- Scheduling
- Flow over time
- ...

Timetable Combinatorial Optimization

Timetable Combinatorial Optimization

Combinatorial Optimization (CO)

Timetable Combinatorial Optimization

Combinatorial Optimization (CO)

Input:

- Set system $F = (A, \mathcal{X})$ (with $\mathcal{X} \subset 2^A$),
- Weights $w_a \in \mathbb{Z}_+$ for every $a \in A$.

Timetable Combinatorial Optimization

Combinatorial Optimization (CO)

Input:

- Set system $F = (A, \mathcal{X})$ (with $\mathcal{X} \subset 2^A$),
- Weights $w_a \in \mathbb{Z}_+$ for every $a \in A$.

Problem: Find $S^* \in \mathcal{X}$ maximizing $w(S) = \sum_{a \in S} w_a$.

Timetable Combinatorial Optimization

Combinatorial Optimization (CO)

Input:

- Set system $F = (A, \mathcal{X})$ (with $\mathcal{X} \subset 2^A$),
- Weights $w_a \in \mathbb{Z}_+$ for every $a \in A$.

Problem: Find $S^* \in \mathcal{X}$ maximizing $w(S) = \sum_{a \in S} w_a$.

Timetable Combinatorial Optimization (TCO)

Timetable Combinatorial Optimization

Combinatorial Optimization (CO)

Input:

- Set system $F = (A, \mathcal{X})$ (with $\mathcal{X} \subset 2^A$),
- Weights $w_a \in \mathbb{Z}_+$ for every $a \in A$.

Problem: Find $S^* \in \mathcal{X}$ maximizing $w(S) = \sum_{a \in S} w_a$.

Timetable Combinatorial Optimization (TCO)

Input:

- A CO instance $F = (A, \mathcal{X}), w$.
- *Durations* $\tau_a \in \mathbb{Z}_+$ for every $a \in A$.
- *Total duration* $T \in \mathbb{Z}_+$.

Timetable Combinatorial Optimization

Combinatorial Optimization (CO)

Input:

- Set system $F = (A, \mathcal{X})$ (with $\mathcal{X} \subset 2^A$),
- Weights $w_a \in \mathbb{Z}_+$ for every $a \in A$.

Problem: Find $S^* \in \mathcal{X}$ maximizing $w(S) = \sum_{a \in S} w_a$.

Timetable Combinatorial Optimization (TCO)

Input:

- A CO instance $F = (A, \mathcal{X}), w$.
- *Durations* $\tau_a \in \mathbb{Z}_+$ for every $a \in A$.
- *Total duration* $T \in \mathbb{Z}_+$.

Problem: Find $S_1, \cdots, S_T \in \mathcal{X}$ satisfying the *duration property* and maximizing $w(S_1) + \cdots + w(S_T)$.

Timetable Combinatorial Optimization

Timetable Combinatorial Optimization (TCO)

Input:

- A CO instance $F = (A, \mathcal{X}), w$.
- *Durations* $\tau_a \in \mathbb{Z}_+$ for every $a \in A$.
- *Total duration* $T \in \mathbb{Z}_+$.

Problem: Find $S_1, \cdots, S_T \in \mathcal{X}$ satisfying the *duration property* and maximizing $w(S_1) + \cdots + w(S_T)$.

Timetable Combinatorial Optimization

Timetable Combinatorial Optimization (TCO)

Input:

- A CO instance $F = (A, \mathcal{X}), w$.
- *Durations* $\tau_a \in \mathbb{Z}_+$ for every $a \in A$.
- *Total duration* $T \in \mathbb{Z}_+$.

Problem: Find $S_1, \cdots, S_T \in \mathcal{X}$ satisfying the *duration property* and maximizing $w(S_1) + \cdots + w(S_T)$.

duration property w.r.t. $a \in A$

$I_a = \{i \in [T] : a \in S_i\}$ is a disjoint union of intervals of length τ_a.

Timetable Combinatorial Optimization

Timetable Combinatorial Optimization (TCO)

Input:

- A CO instance $F = (A, \mathcal{X}), w$.
- *Durations* $\tau_a \in \mathbb{Z}_+$ for every $a \in A$.
- *Total duration* $T \in \mathbb{Z}_+$.

Problem: Find $S_1, \cdots, S_T \in \mathcal{X}$ satisfying the *duration property* and maximizing $w(S_1) + \cdots + w(S_T)$.

duration property w.r.t. $a \in A$

$I_a = \{i \in [T] : a \in S_i\}$ is a disjoint union of intervals of length τ_a.

duration property

Duration property w.r.t a is satisfied for all $a \in A$.

TCO - Examples

Example: Matchings

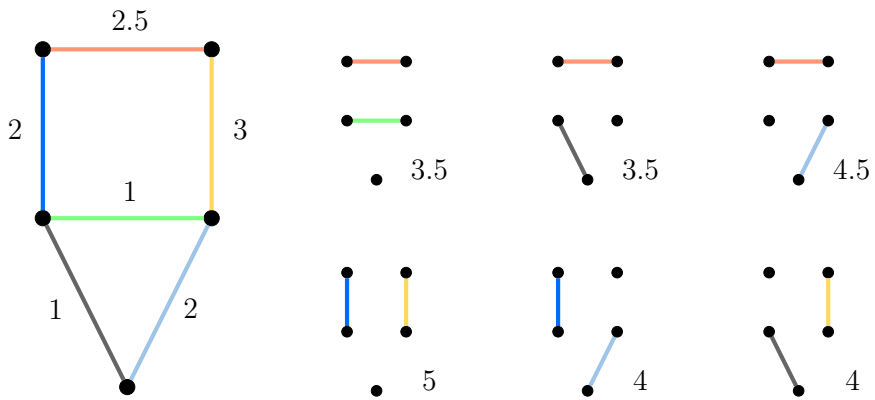
- A is the edge set of a graph $G = (V, A)$
- $\mathcal{X}$ is the set of all matchings of G

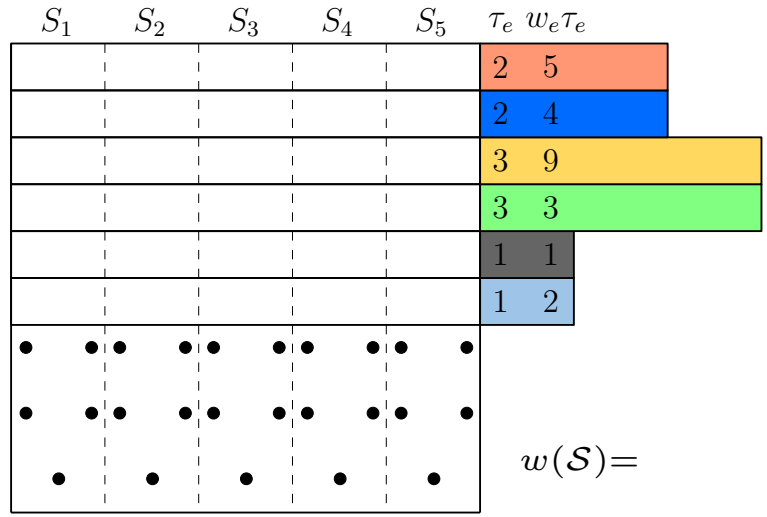
TCO - Examples

Example: Matchings

- A is the edge set of a graph $G = (V, A)$
- $\mathcal{X}$ is the set of all matchings of G

Example: Timetable Matchings



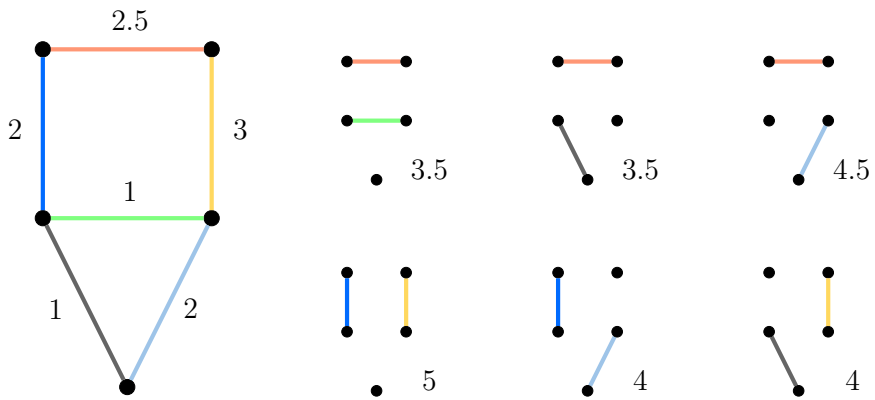


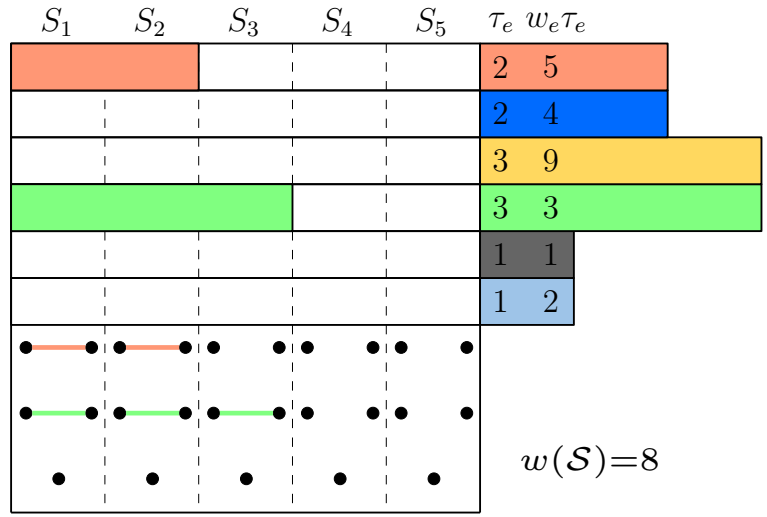
TCO - Examples

Example: Matchings

- A is the edge set of a graph $G = (V, A)$
- $\mathcal{X}$ is the set of all matchings of G

Example: Timetable Matchings



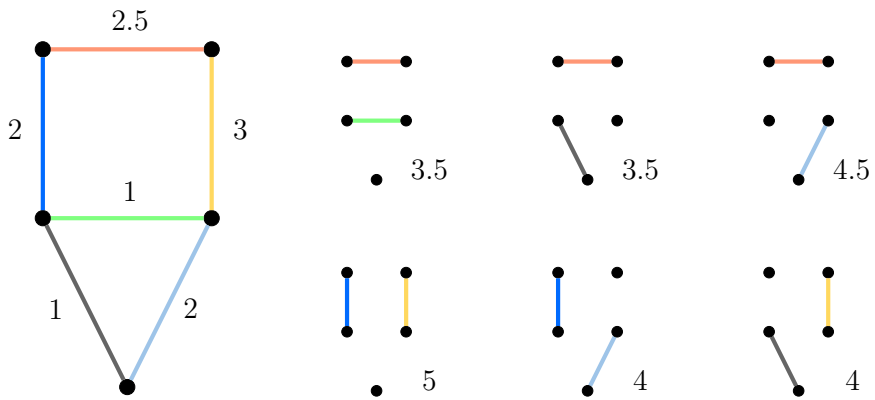


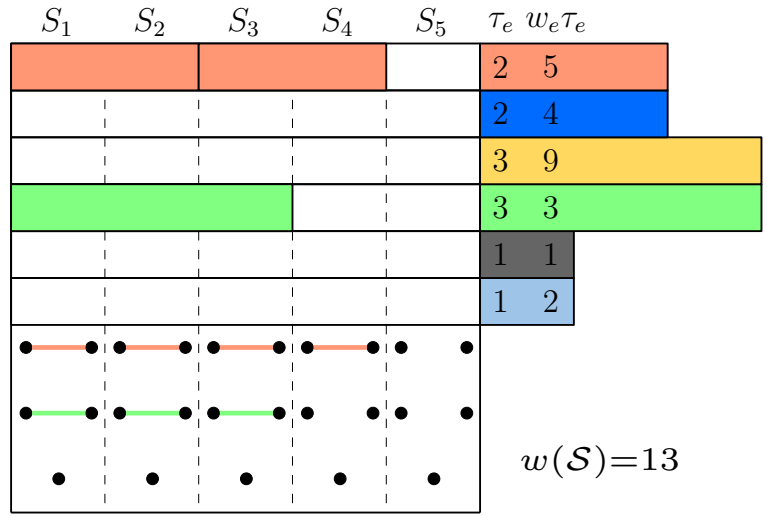
TCO - Examples

Example: Matchings

- A is the edge set of a graph $G = (V, A)$
- $\mathcal{X}$ is the set of all matchings of G

Example: Timetable Matchings



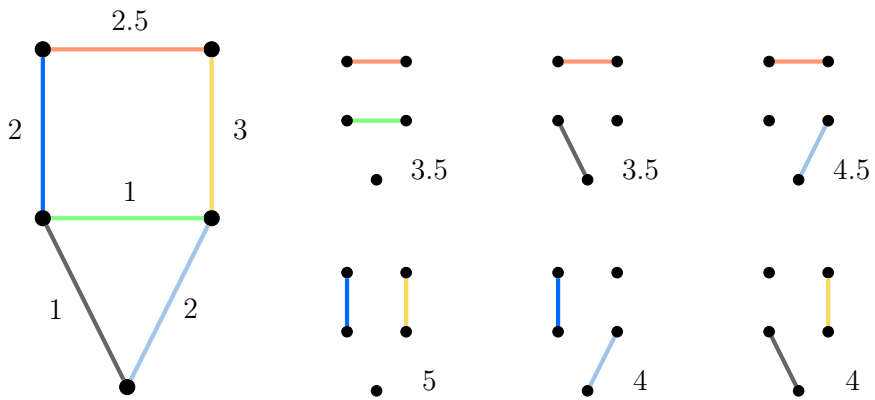


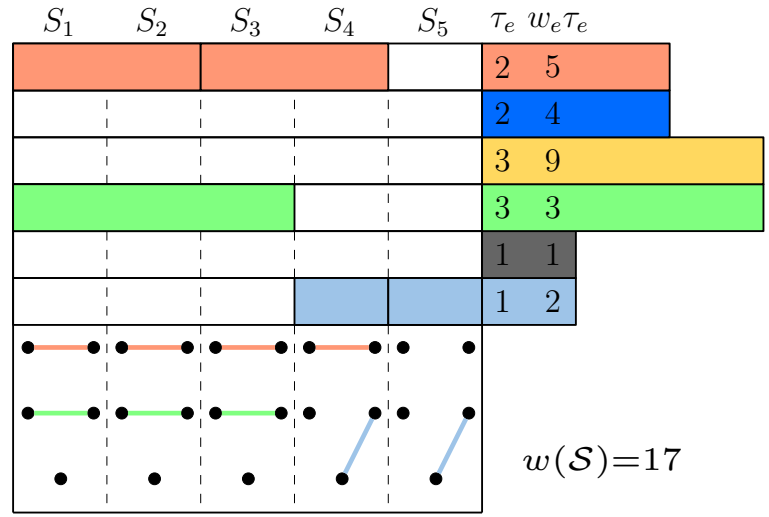
TCO - Examples

Example: Matchings

- A is the edge set of a graph $G = (V, A)$
- $\mathcal{X}$ is the set of all matchings of G

Example: Timetable Matchings



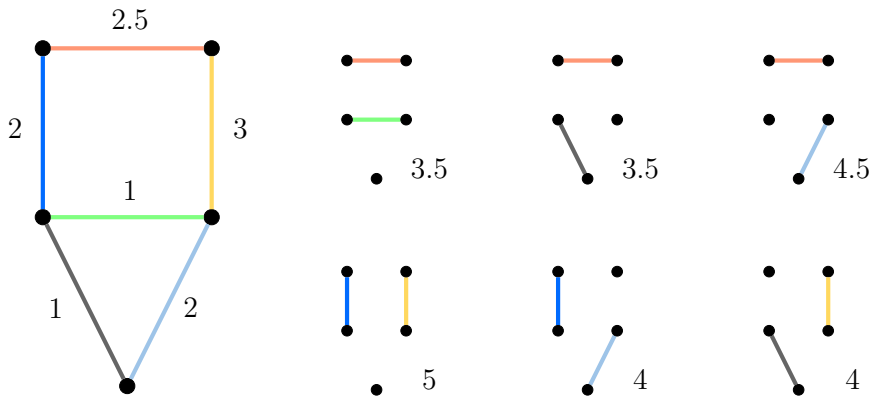


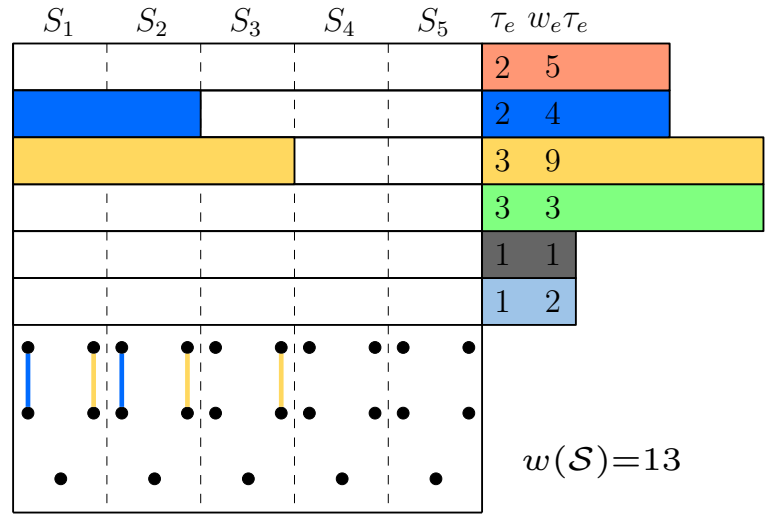
TCO - Examples

Example: Matchings

- A is the edge set of a graph $G = (V, A)$
- $\mathcal{X}$ is the set of all matchings of G

Example: Timetable Matchings



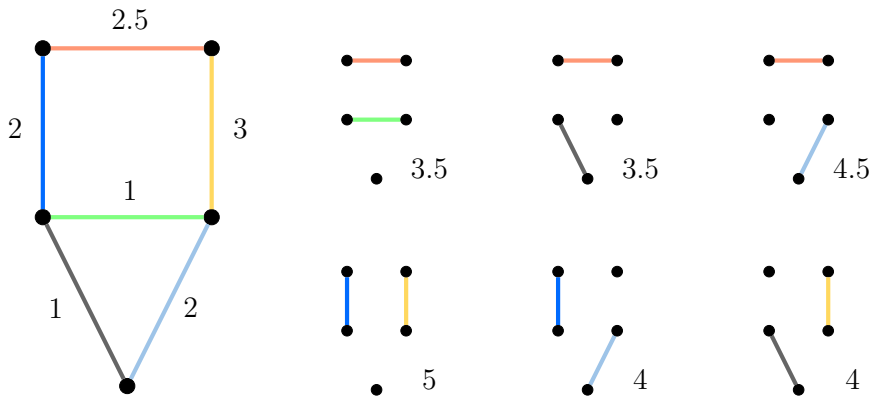


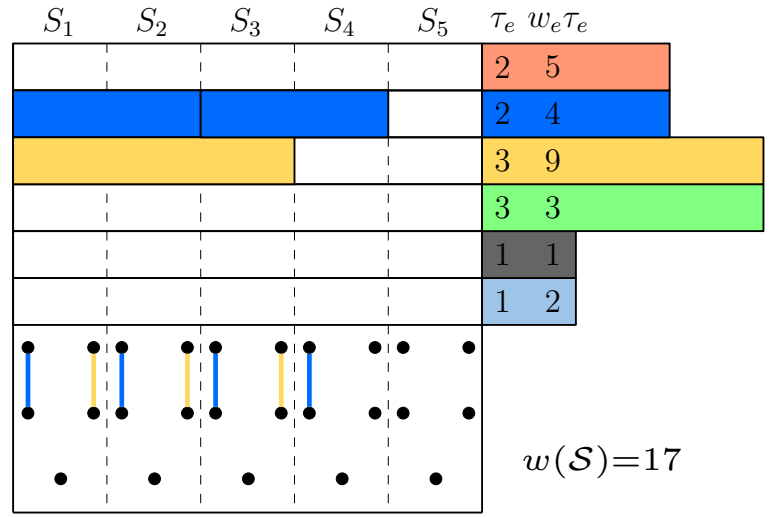
TCO - Examples

Example: Matchings

- A is the edge set of a graph $G = (V, A)$
- $\mathcal{X}$ is the set of all matchings of G

Example: Timetable Matchings



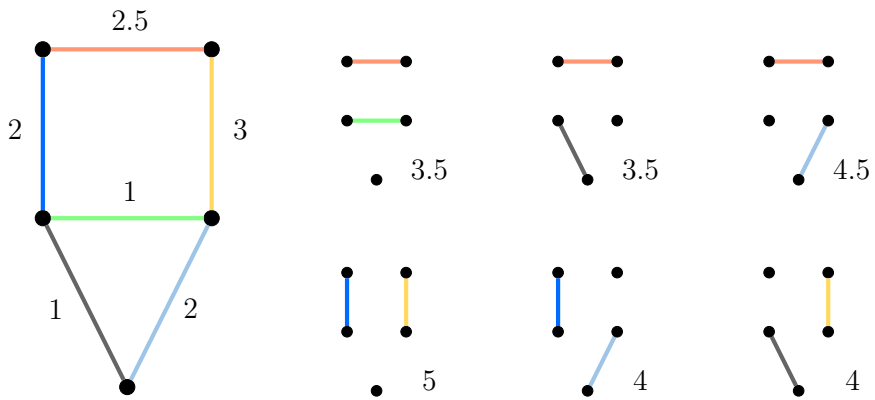


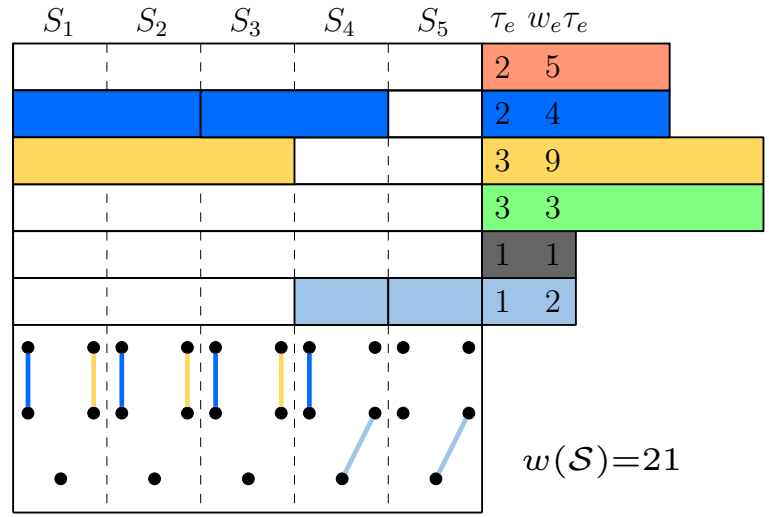
TCO - Examples

Example: Matchings

- A is the edge set of a graph $G = (V, A)$
- $\mathcal{X}$ is the set of all matchings of G

Example: Timetable Matchings



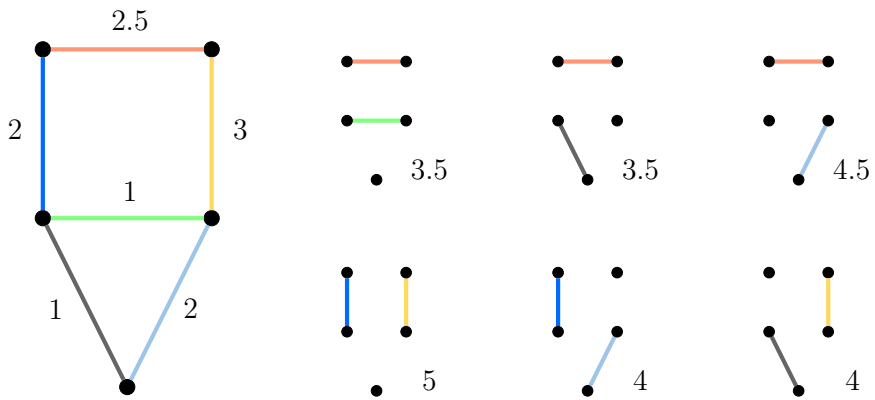


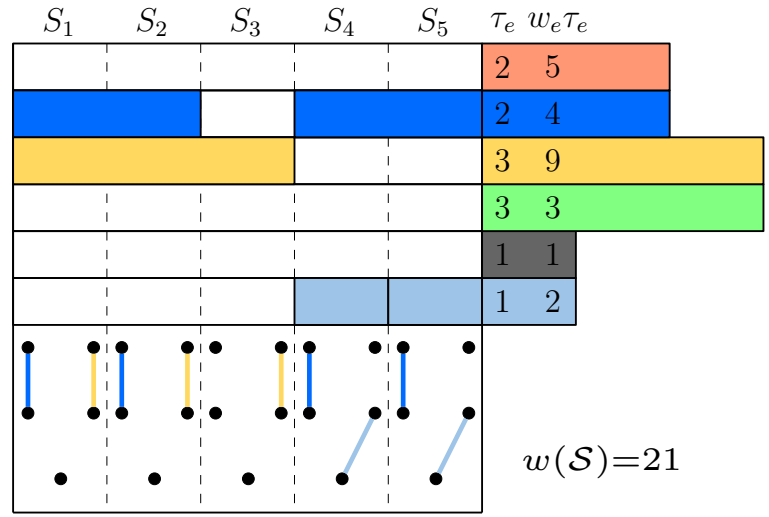
TCO - Examples

Example: Matchings

- A is the edge set of a graph $G = (V, A)$
- $\mathcal{X}$ is the set of all matchings of G

Example: Timetable Matchings



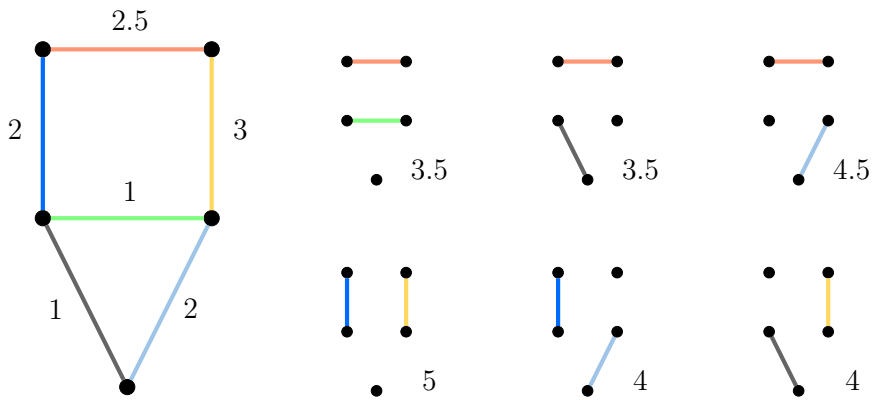


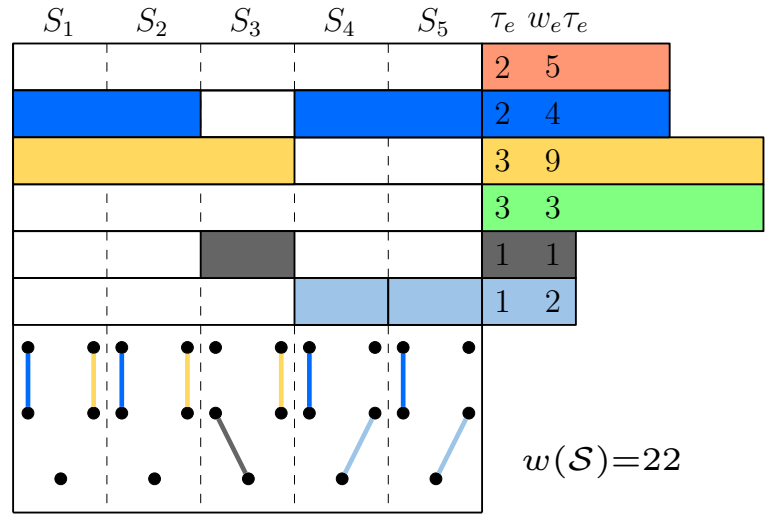
TCO - Examples

Example: Matchings

- A is the edge set of a graph $G = (V, A)$
- $\mathcal{X}$ is the set of all matchings of G

Example: Timetable Matchings



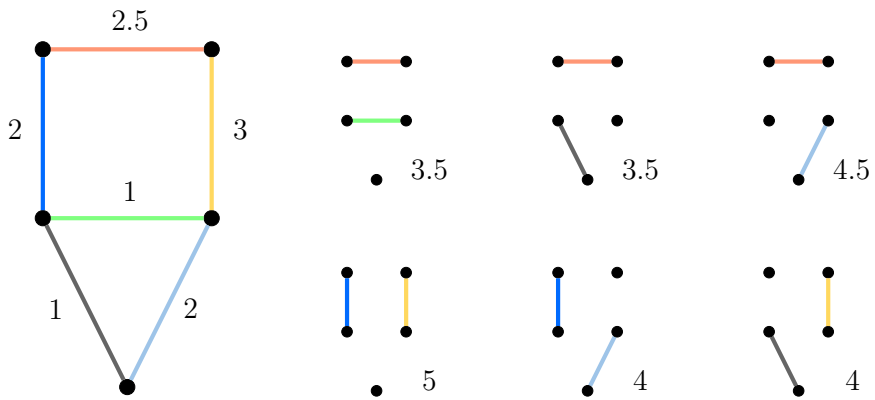


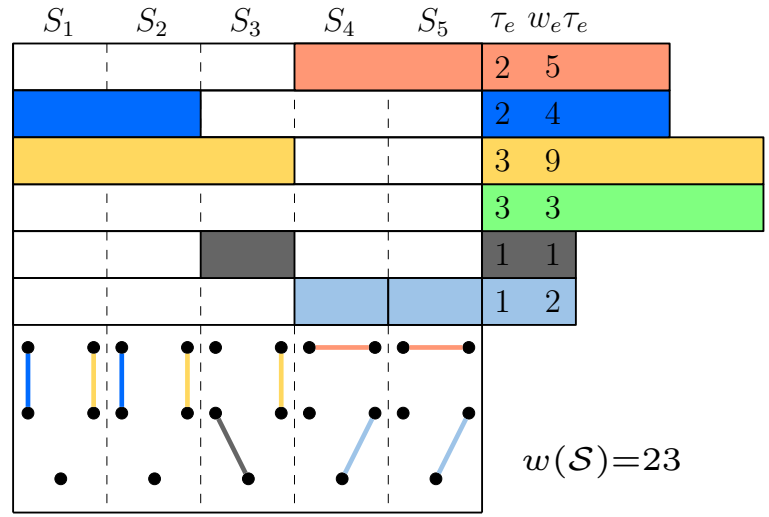
TCO - Examples

Example: Matchings

- A is the edge set of a graph $G = (V, A)$
- $\mathcal{X}$ is the set of all matchings of G

Example: Timetable Matchings





TCO - Examples

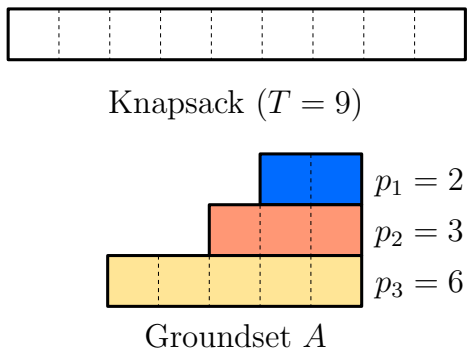
Example: Integer Knapsack

- Let $\mathcal{X} = \{S \subseteq A : |S| \leqslant 1\}$, the rank-1 uniform matroid
- $a \in X_t$ if position t is occupied by an object of type a

TCO - Examples

Example: Integer Knapsack

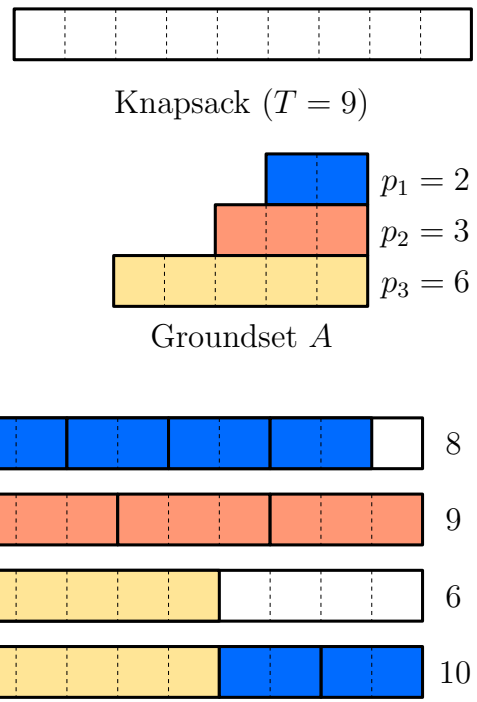
- Let $\mathcal{X} = \{S \subseteq A : |S| \leqslant 1\}$, the rank-1 uniform matroid
- $a \in X_t$ if position t is occupied by an object of type a



TCO - Examples

Example: Integer Knapsack

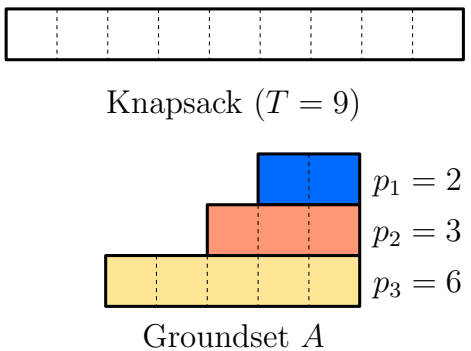
- Let $\mathcal{X} = \{S \subseteq A : |S| \leqslant 1\}$, the rank-1 uniform matroid
- $a \in X_t$ if position t is occupied by an object of type a

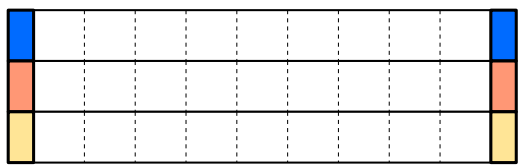


TCO - Examples

Example: Integer Knapsack

- Let $\mathcal{X} = \{S \subseteq A : |S| \leqslant 1\}$, the rank-1 uniform matroid
- $a \in X_t$ if position t is occupied by an object of type a

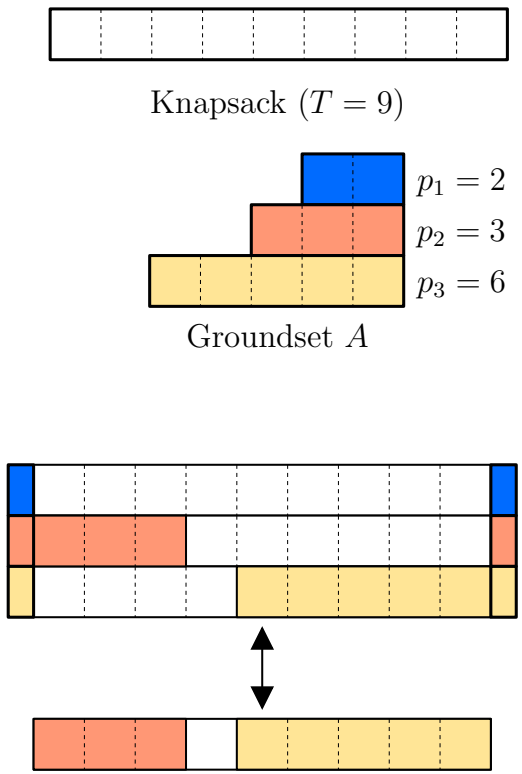




TCO - Examples

Example: Integer Knapsack

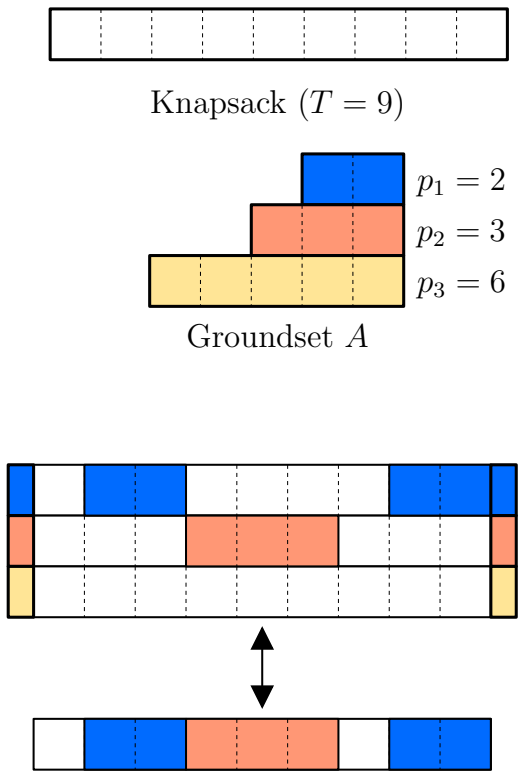
- Let $\mathcal{X} = \{S \subseteq A : |S| \leqslant 1\}$, the rank-1 uniform matroid
- $a \in X_t$ if position t is occupied by an object of type a



TCO - Examples

Example: Integer Knapsack

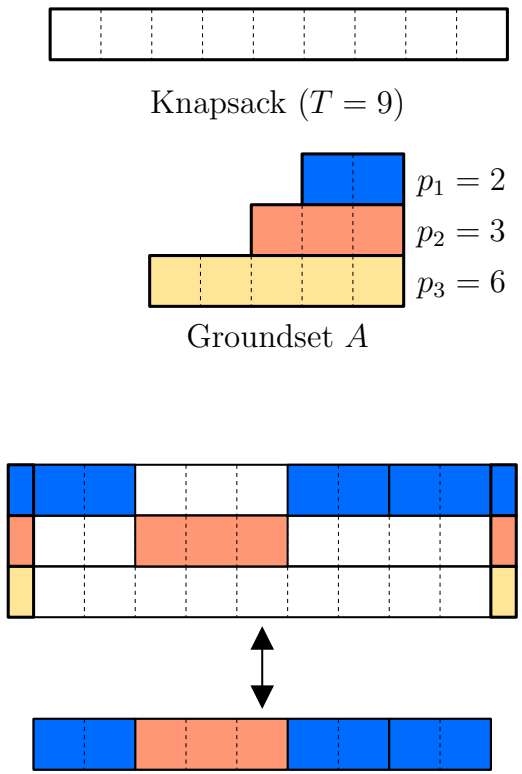
- Let $\mathcal{X} = \{S \subseteq A : |S| \leqslant 1\}$, the rank-1 uniform matroid
- $a \in X_t$ if position t is occupied by an object of type a



TCO - Examples

Example: Integer Knapsack

- Let $\mathcal{X} = \{S \subseteq A : |S| \leqslant 1\}$, the rank-1 uniform matroid
- $a \in X_t$ if position t is occupied by an object of type a



TCO - Examples

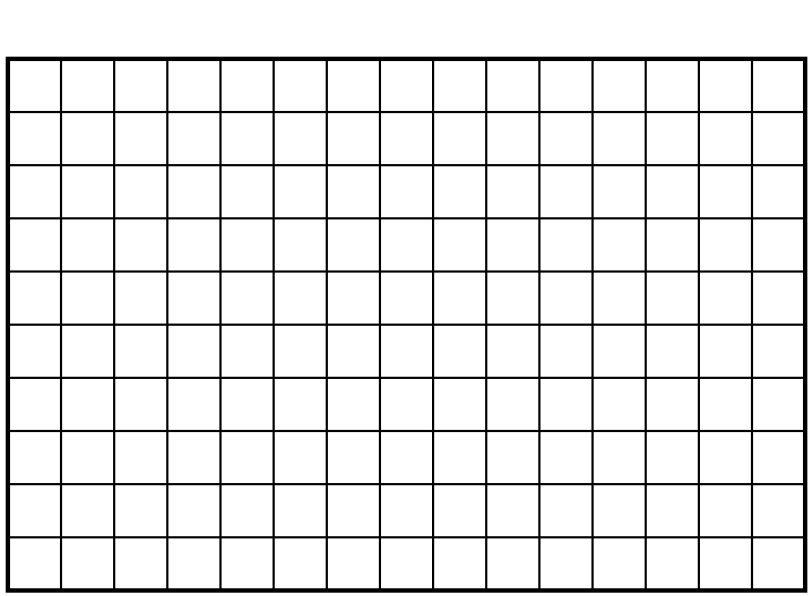
Example: Orthogonal Retangle Packing

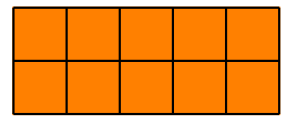
- $\mathcal{X} = \{$collections of intervals that do not overlap$\}$.

TCO - Examples

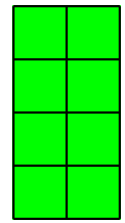
Example: Orthogonal Retangle Packing

- $\mathcal{X} = \{$collections of intervals that do not overlap$\}$.

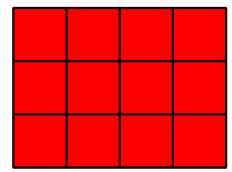




$w_1 = 3$



$w_2 = 5$

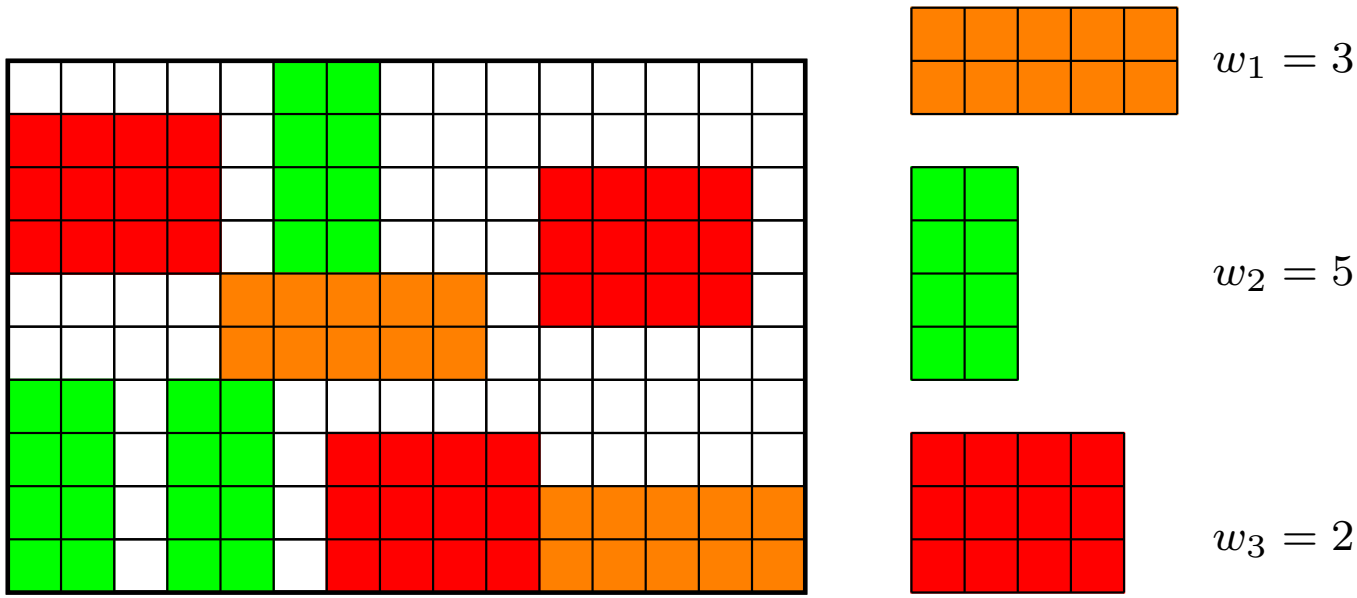


$w_3 = 2$

TCO - Examples

Example: Orthogonal Retangle Packing

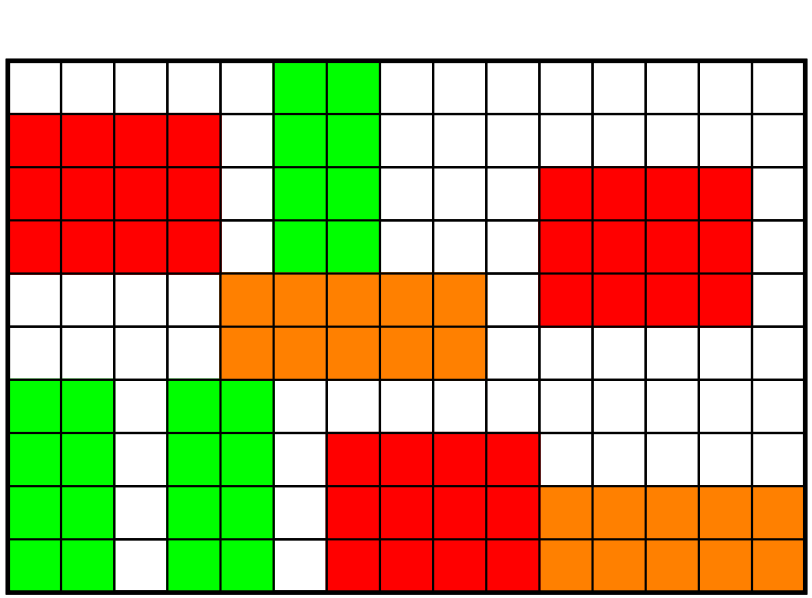
- $\mathcal{X} = \{\text{collections of intervals that do not overlap}\}$.

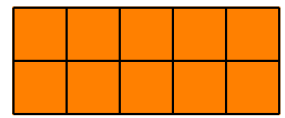


TCO - Examples

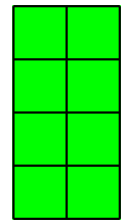
Example: Orthogonal Retangle Packing

- $\mathcal{X} = \{\text{collections of intervals that do not overlap}\}$.

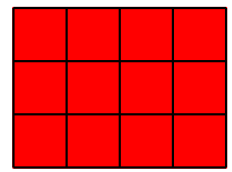




$w_1 = 3$



$w_2 = 5$



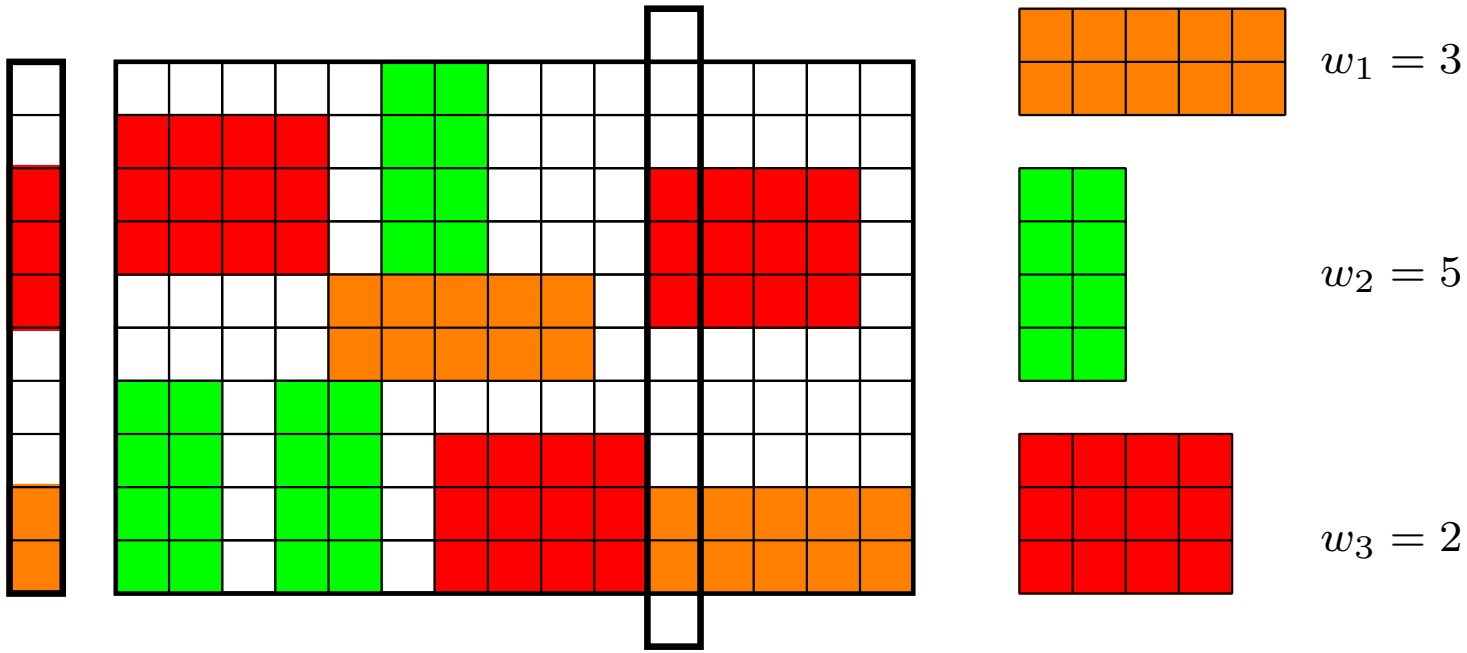
$w_3 = 2$

$$w = 27$$

TCO - Examples

Example: Orthogonal Retangle Packing

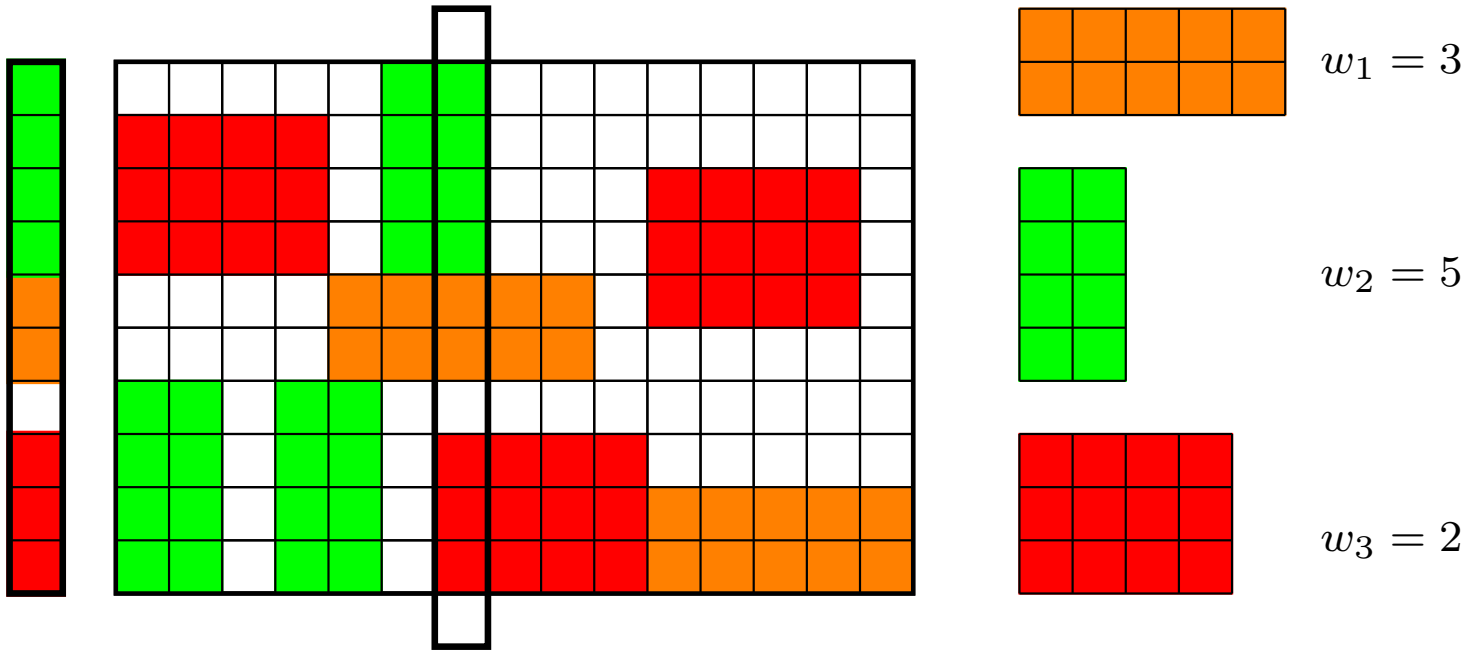
- $\mathcal{X} = \{$collections of intervals that do not overlap$\}$.



TCO - Examples

Example: Orthogonal Retangle Packing

- $\mathcal{X} = \{$collections of intervals that do not overlap$\}$.



Results for TCO

Results for TCO

Complexity Results:

- Timetable Matroid Optimization is NP-hard.
- Timetable Matchings is APX-hard.
- ...

Results for TCO

Complexity Results:

- Timetable Matroid Optimization is NP-hard.
- Timetable Matchings is APX-hard.
- ...

Approximation Algorithm:

Theorem 1

If $(A, \mathcal{X})$ are independence systems and there is β-approximation algorithm for the maximization problem, then there exists a $\alpha\beta$-approximation algorithm for the timetable counterpart, with $\alpha \approx 1.691$.

Results for TCO

Complexity Results:

- Timetable Matroid Optimization is NP-hard.
- Timetable Matchings is APX-hard.
- ...

Approximation Algorithm:

Theorem 1

If $(A, \mathcal{X})$ are independence systems and there is β-approximation algorithm for the maximization problem, then there exists a $\alpha\beta$-approximation algorithm for the timetable counterpart, with $\alpha \approx 1.691$.

Ingredients

- Simple algorithm based on black-box execution of β-approximation.

Results for TCO

Complexity Results:

- Timetable Matroid Optimization is NP-hard.
- Timetable Matchings is APX-hard.
- ...

Approximation Algorithm:

Theorem 1

If $(A, \mathcal{X})$ are independence systems and there is β-approximation algorithm for the maximization problem, then there exists a $\alpha\beta$-approximation algorithm for the timetable counterpart, with $\alpha \approx 1.691$.

Ingredients

- Simple algorithm based on black-box execution of β-approximation.
- Combinatorial argument to bounds approximation guarantee for every T separately by a LP.

Results for TCO

Complexity Results:

- Timetable Matroid Optimization is NP-hard.
- Timetable Matchings is APX-hard.
- ...

Approximation Algorithm:

Theorem 1

If $(A, \mathcal{X})$ are independence systems and there is β-approximation algorithm for the maximization problem, then there exists a $\alpha\beta$-approximation algorithm for the timetable counterpart, with $\alpha \approx 1.691$.

Ingredients

- Simple algorithm based on black-box execution of β-approximation.
- Combinatorial argument to bounds approximation guarantee for every T separately by a LP.
- Infinite LP bounding all previous LPs.

Results for TCO

Complexity Results:

- Timetable Matroid Optimization is NP-hard.
- Timetable Matchings is APX-hard.
- ...

Approximation Algorithm:

Theorem 1

If $(A, \mathcal{X})$ are independence systems and there is β-approximation algorithm for the maximization problem, then there exists a $\alpha\beta$-approximation algorithm for the timetable counterpart, with $\alpha \approx 1.691$.

Ingredients

- Simple algorithm based on black-box execution of β-approximation.
- Combinatorial argument to bounds approximation guarantee for every T separately by a LP.
- Infinite LP bounding all previous LPs.
- Explicit construction of a solution with value α for dual LP.

Results for TCO

Complexity Results:

- Timetable Matroid Optimization is NP-hard.
- Timetable Matchings is APX-hard.
- ...

Approximation Algorithm:

Theorem 1

If $(A, \mathcal{X})$ are independence systems and there is β-approximation algorithm for the maximization problem, then there exists a $\alpha\beta$-approximation algorithm for the timetable counterpart, with $\alpha \approx 1.691$.

Paper:

A.D., Bosio S. and Weismantel R. **Timetable combinatorial optimization: a complexity and approximability study**. *Submitted*.

TCO with Upper Bounds

TCO with Upper Bounds

Limitations of TCO:

Can one model Binary Knapsack as a TCO?

TCO with Upper Bounds

Limitations of TCO:

Can one model Binary Knapsack as a TCO? No..

TCO with Upper Bounds

Limitations of TCO:

Can one model Binary Knapsack as a TCO? No..

Bounded TCO (BTCO):

- The input also specifies upper bounds $c_a \in \mathbb{Z}_+$ for every $a \in A$.

TCO with Upper Bounds

Limitations of TCO:

Can one model Binary Knapsack as a TCO? No..

Bounded TCO (BTCO):

- The input also specifies upper bounds $c_a \in \mathbb{Z}_+$ for every $a \in A$.
- Additional requirement on $S = (S_1, \cdots, S_T)$: The sets $I_a = \{i \in [T] : a \in S_i\}$ satisfy

$$|I_a| \leqslant c_a \tau_a.$$

TCO with Upper Bounds

Limitations of TCO:

Can one model Binary Knapsack as a TCO? No..

Bounded TCO (BTCO):

- The input also specifies upper bounds $c_a \in \mathbb{Z}_+$ for every $a \in A$.
- Additional requirement on $S = (S_1, \cdots, S_T)$: The sets $I_a = \{i \in [T] : a \in S_i\}$ satisfy

$$|I_a| \leqslant c_a \tau_a.$$

Binary Knapsack: repeat TCO reduction for IK and set $c_a = 1$ for all elements.

Results for BTCO

Results for BTCO

Complexity Results:

- Binary Timetable Maximum Matroid Basis is strongly NP-hard.
- Binary Timetable Maximum Bipartite Matching is APX-hard even when $T = 2$.

Results for BTCO

Complexity Results:

- Binary Timetable Maximum Matroid Basis is strongly NP-hard.
- Binary Timetable Maximum Bipartite Matching is APX-hard even when $T = 2$.

Approximation Algorithms:

Theorem 2

The Binary Timetable Maximum Matroid Basis problem admits a 4-approximation algorithm.

Results for BTCO

Complexity Results:

- Binary Timetable Maximum Matroid Basis is strongly NP-hard.
- Binary Timetable Maximum Bipartite Matching is APX-hard even when $T = 2$.

Approximation Algorithms:

Theorem 2

The Binary Timetable Maximum Matroid Basis problem admits a 4-approximation algorithm.

Theorem 3

The Binary Timetable Maximum Spanning Forest problem admits a 3-approximation algorithm.

Results for BTCO

Complexity Results:

- Binary Timetable Maximum Matroid Basis is strongly NP-hard.
- Binary Timetable Maximum Bipartite Matching is APX-hard even when $T = 2$.

Approximation Algorithms:

Theorem 2

The Binary Timetable Maximum Matroid Basis problem admits a 4-approximation algorithm.

Theorem 3

The Binary Timetable Maximum Spanning Forest problem admits a 3-approximation algorithm.

Paper:

A.D., Bosio S., Weismantel R. and Zenklusen R. **Timetable matroid optimization**. *Technical Report*.

Theorem 3 - Proof Idea (1)

Theorem 3 - Proof Idea (1)

Graph Balancing:

Given a graph $G = (V, E)$ and edge weights w_e, obtain a direction $D = (V, E')$ of G so as to minimize

$$\max_{v \in V} \sum_{e \in \delta^+(v)} w_e.$$

Theorem 3 - Proof Idea (1)

Graph Balancing:

Given a graph $G = (V, E)$ and edge weights w_e, obtain a direction $D = (V, E')$ of G so as to minimize

$$\max_{v \in V} \sum_{e \in \delta^+(v)} w_e.$$

IP1

$$\min \max_{v \in V} \sum_{e \in E_v} \tau_e x_v^e$$

$$x_u^e + x_v^e = 1 \qquad \forall e = (u, v) \in E$$

$$x_v^e \in \{0, 1\} \qquad \forall e \in E, v \in e.$$

Theorem 3 - Proof Idea (1)

Graph Balancing:

Given a graph $G = (V, E)$ and edge weights w_e, obtain a direction $D = (V, E')$ of G so as to minimize

$$\max_{v \in V} \sum_{e \in \delta^+(v)} w_e.$$

IP1

$$\min \max_{v \in V} \sum_{e \in E_v} \tau_e x_v^e$$

$$x_u^e + x_v^e = 1 \qquad \forall e = (u, v) \in E$$

$$x_v^e \in \{0, 1\} \qquad \forall e \in E, v \in e.$$

Ebenlendr, Krčál and Sgall (2007):

- NP-hard to approximate within $1.5 - \epsilon$.
- Integrality gap of IP1 is 2.
- A 1.75-approximation algorithm.

Theorem 3 - Proof Idea (2)

Theorem 3 - Proof Idea (2)

Capacitated Graph Balancing:

- Strict bound of load at every vertex.
- Not all edges must be directed.
- Maximize weight of directed edges.

Theorem 3 - Proof Idea (2)

Capacitated Graph Balancing:

- Strict bound of load at every vertex.
- Not all edges must be directed.
- Maximize weight of directed edges.

IP2

$$\max \sum_{e=\{u,v\}\in E} \tau_e w_e (x_u^e + x_v^e) \tag{1}$$

$$x_u^e + x_v^e \leqslant 1 \qquad \forall e = \{u,v\} \in E \tag{2}$$

$$\sum_{e \in E_v} \tau_e x_v^e \leqslant T \qquad \forall v \in V \tag{3}$$

$$x_u^e, x_v^e \in \{0,1\} \qquad \forall e = \{u,v\} \in E. \tag{4}$$

Theorem 3 - Proof Idea (2)

Capacitated Graph Balancing:

- Strict bound of load at every vertex.
- Not all edges must be directed.
- Maximize weight of directed edges.

IP2

$$\max \sum_{e=\{u,v\}\in E} \tau_e w_e (x_u^e + x_v^e) \tag{1}$$

$$x_u^e + x_v^e \leqslant 1 \qquad \forall e = \{u,v\} \in E \tag{2}$$

$$\sum_{e\in E_v} \tau_e x_v^e \leqslant T \qquad \forall v \in V \tag{3}$$

$$x_u^e, x_v^e \in \{0,1\} \qquad \forall e = \{u,v\} \in E. \tag{4}$$

Phase1: Solve LP2, the LP relaxation of IP2

Theorem 3 - Proof Idea (2)

Capacitated Graph Balancing:

- Strict bound of load at every vertex.
- Not all edges must be directed.
- Maximize weight of directed edges.

IP2

$$\max \sum_{e=\{u,v\}\in E} \tau_e w_e (x_u^e + x_v^e) \tag{1}$$

$$x_u^e + x_v^e \leqslant 1 \qquad \forall e = \{u,v\} \in E \tag{2}$$

$$\sum_{e\in E_v} \tau_e x_v^e \leqslant T \qquad \forall v \in V \tag{3}$$

$$x_u^e, x_v^e \in \{0,1\} \qquad \forall e = \{u,v\} \in E. \tag{4}$$

Phase1: Solve LP2, the LP relaxation of IP2 $\Rightarrow \bar{x}$.

Theorem 3 - Proof Idea (3)

Theorem 3 - Proof Idea (3)

Properties of IP2

- Feasible solutions to IP2 $\not\Leftrightarrow$ Sets of edges packable in a timetable.

Theorem 3 - Proof Idea (3)

Properties of IP2

- Feasible solutions to IP2 $\not\Leftrightarrow$ Sets of edges packable in a timetable.
- However,

Theorem 3 - Proof Idea (3)

Properties of IP2

- Feasible solutions to IP2 $\not\Leftrightarrow$ Sets of edges packable in a timetable.
- However,

Lemma 1

Any solution to IP2 can be efficiently transformed into a timetable forest, incurring a loss of at most a factor 2.

Theorem 3 - Proof Idea (3)

Properties of IP2

- Feasible solutions to IP2 $\not\Leftrightarrow$ Sets of edges packable in a timetable.
- However,

Lemma 1

Any solution to IP2 can be efficiently transformed into a timetable forest, incurring a loss of at most a factor 2.

Lemma 2

$OPT_{LP2} \geqslant OPT$.

Theorem 3 - Proof Idea (3)

Properties of IP2

- Feasible solutions to IP2 $\not\Leftrightarrow$ Sets of edges packable in a timetable.
- However,

Lemma 1

Any solution to IP2 can be efficiently transformed into a timetable forest, incurring a loss of at most a factor 2.

Lemma 2

$OPT_{LP2} \geqslant OPT$.

Additionally:

Lemma 3

$\bar{x}$ can be efficiently rounded to a solution $\hat{x}$ with the properties

- The collection of fractional edges E_f in $\hat{x}$ is a pseudoforest (trees and trees with an additional edge).
- Is every component of E_f at most one edge $e = (u, v)$ satisfies $\hat{x}_u^e + \hat{x}_v^e \in (0, 1)$.

Theorem 3 - Proof Idea (3)

Properties of IP2

- Feasible solutions to IP2 $\not\Leftrightarrow$ Sets of edges packable in a timetable.
- However,

Lemma 1

Any solution to IP2 can be efficiently transformed into a timetable forest, incurring a loss of at most a factor 2.

Lemma 2

$OPT_{LP2} \geqslant OPT$.

Additionally:

Lemma 3

$\bar{x}$ can be efficiently rounded to a solution $\hat{x}$ with the properties

- The collection of fractional edges E_f in $\hat{x}$ is a pseudoforest (trees and trees with an additional edge).
- Is every component of E_f at most one edge $e = (u, v)$ satisfies $\hat{x}_u^e + \hat{x}_v^e \in (0, 1)$.

Phase2: Perform the rounding and obtain $\hat{x}$.

Theorem 3 - Proof Idea (4)

Theorem 3 - Proof Idea (4)

A 3.5-approximation algorithm (Phase3):

Theorem 3 - Proof Idea (4)

A 3.5-approximation algorithm (Phase3):

- Partition the support of $\hat{x}$ into the integral part E_I and the fractional part E_f.

Theorem 3 - Proof Idea (4)

A 3.5-approximation algorithm (Phase3):

- Partition the support of $\hat{x}$ into the integral part E_I and the fractional part E_f.
- If $\text{Weight}(E_I) \geqslant \frac{4}{7}\text{Weight}(\hat{x})$ use Lemma 1.

Theorem 3 - Proof Idea (4)

A 3.5-approximation algorithm (Phase3):

- Partition the support of $\hat{x}$ into the integral part E_I and the fractional part E_f.
- If $\text{Weight}(E_I) \geqslant \frac{4}{7}\text{Weight}(\hat{x})$ use Lemma 1.
- Otherwise $\text{Weight}(E_f) \geqslant \frac{3}{7}\text{Weight}(\hat{x})$.

Theorem 3 - Proof Idea (4)

A 3.5-approximation algorithm (Phase3):

- Partition the support of $\hat{x}$ into the integral part E_I and the fractional part E_f.
- If $\text{Weight}(E_I) \geqslant \frac{4}{7}\text{Weight}(\hat{x})$ use Lemma 1.
- Otherwise $\text{Weight}(E_f) \geqslant \frac{3}{7}\text{Weight}(\hat{x})$.
- Pack a $\frac{2}{3}$ fraction of E_f into a timetable, by removing at most one edge from each cycle.

Theorem 3 - Proof Idea (4)

A 3.5-approximation algorithm (Phase3):

- Partition the support of $\hat{x}$ into the integral part E_I and the fractional part E_f.
- If $\text{Weight}(E_I) \geqslant \frac{4}{7}\text{Weight}(\hat{x})$ use Lemma 1.
- Otherwise $\text{Weight}(E_f) \geqslant \frac{3}{7}\text{Weight}(\hat{x})$.
- Pack a $\frac{2}{3}$ fraction of E_f into a timetable, by removing at most one edge from each cycle.

To obtain a factor 3 one needs to work a bit more...

Conclusions and Future Work

Conclusions and Future Work

We've seen

- A new way to incorporate a temporal dimension into CO problems.

Conclusions and Future Work

We've seen

- A new way to incorporate a temporal dimension into CO problems.
- Connections to various scheduling and packing problems.

Conclusions and Future Work

We've seen

- A new way to incorporate a temporal dimension into CO problems.
- Connections to various scheduling and packing problems.
- Combinatorial and LP-based algorithms.

Conclusions and Future Work

We've seen

- A new way to incorporate a temporal dimension into CO problems.
- Connections to various scheduling and packing problems.
- Combinatorial and LP-based algorithms.

We hope to

- Improve approximation algorithms.

Conclusions and Future Work

We've seen

- A new way to incorporate a temporal dimension into CO problems.
- Connections to various scheduling and packing problems.
- Combinatorial and LP-based algorithms.

We hope to

- Improve approximation algorithms.
- Obtain algorithms for Binary timetable independence systems problems (Matching...).

Conclusions and Future Work

We've seen

- A new way to incorporate a temporal dimension into CO problems.
- Connections to various scheduling and packing problems.
- Combinatorial and LP-based algorithms.

We hope to

- Improve approximation algorithms.
- Obtain algorithms for Binary timetable independence systems problems (Matching...).
- Understand the complexity of Timetable Matroid problems.

Conclusions and Future Work

We've seen

- A new way to incorporate a temporal dimension into CO problems.
- Connections to various scheduling and packing problems.
- Combinatorial and LP-based algorithms.

We hope to

- Improve approximation algorithms.
- Obtain algorithms for Binary timetable independence systems problems (Matching...).
- Understand the complexity of Timetable Matroid problems.

Thank You!