Robust Discrete Optimization Under Ellipsoidal Uncertainty

Chifaa Dahik

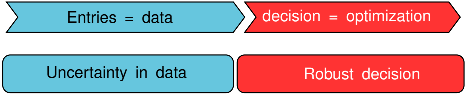
Jean-Marc Nicod, Landy Rabehasaina and Zeina Al Masry

Université de Bourgogne Franche-Comté

December, 3 2021. Workshop SCALE GDR RO Lyon

Need for robustness

Context



Entries = data

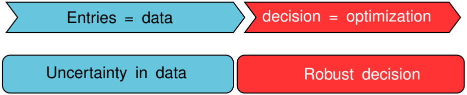
decision = optimization

Uncertainty in data

Robust decision

Need for robustness

Context



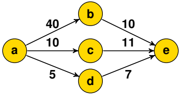
Entries = data

decision = optimization

Uncertainty in data

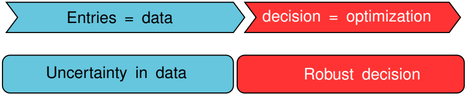
Robust decision

Situation example



Need for robustness

Context



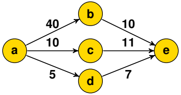
Entries = data

decision = optimization

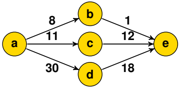
Uncertainty in data

Robust decision

Situation example



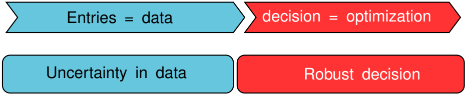
Scenario 1



Scenario 2

Need for robustness

Context



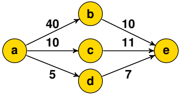
Entries = data

decision = optimization

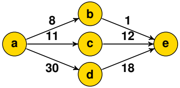
Uncertainty in data

Robust decision

Situation example



Scenario 1



Scenario 2

	Scenario 1	Scenario 2	Worst Scenario
a → b → e	50 min	9 min	50 min
a → c → e	21 min	23 min	23 min
a → d → e	12 min	48 min	48 min

Need for robustness

Resulting facts

⇨ Uncertainty exists

Need for robustness

Resulting facts

- Uncertainty exists
- Considering it is important

Need for robustness

Resulting facts

- Uncertainty exists
- Considering it is important
- It makes problems harder

Need for robustness

Resulting facts

- Uncertainty exists
- Considering it is important
- It makes problems harder
- Optimal solutions are not valid

Uncertainty definition

Definition, cause, available information

➡ Uncertainty= *epistemic situation with unknown or imperfect information*

Uncertainty definition

Definition, cause, available information

➡ Uncertainty= *epistemic situation with unknown or imperfect information*

➡ It can be caused by future events, physical measurements that are already made, or the unknown

Uncertainty definition

Definition, cause, available information

➡ Uncertainty= *epistemic situation with unknown or imperfect information*

➡ It can be caused by future events, physical measurements that are already made, or the unknown

➡ Available information or assumptions can exist: probability distribution, belonging to a set, information from the past, etc.

Uncertainty definition

Definition, cause, available information

- Uncertainty= *epistemic situation with unknown or imperfect information*
- It can be caused by future events, physical measurements that are already made, or the unknown
- Available information or assumptions can exist: probability distribution, belonging to a set, information from the past, etc.
- Uncertainty in an optimization problem:

$$\min_{x \in X_d} f(x, d),$$

X_d: feasible set under the realization d.

Chosen model of uncertainty

- Different approaches exist
- The choice of the approach is made for many reasons: how close to representing the reality, how easy the problem is to solve

Chosen model of uncertainty

Different definitions of worst case based approaches

Absolute robust decision

Chosen model of uncertainty

Different definitions of worst case based approaches

Absolute robust decision

[Kouvelis *et al.* 2004, Buchheim and Kurtz 2018]

$$\min_{x \in X_d} f(x, d)$$

$$z_A = \min_{x \in \cap_{d \in U} X_d} \max_{d \in U} f(x, d)$$

U is an uncertainty set
Best in worst case, with cost=objective

Chosen model of uncertainty

Different definitions of worst case based approaches

Absolute robust decision

Robust deviation decision

Chosen model of uncertainty

Different definitions of worst case based approaches

Absolute robust decision

Robust deviation decision

[Kouvelis *et al.* 2004, Buchheim and Kurtz 2018]

$$\min_{x \in X_d} f(x, d)$$

$$z_D = \min_{x \in \cap_{d \in U} X_d} \max_{d \in U} \left(f(x, d) - f(x_d^*, d) \right)$$

Best in worst case, with cost=deviation from optimal

Chosen model of uncertainty

Different definitions of worst case based approaches

Absolute robust decision

Robust deviation decision

Relative robust decision

Chosen model of uncertainty

Different definitions of worst case based approaches

Absolute robust decision

Robust deviation decision

Relative robust decision

[Kouvelis *et al.* 2004, Buchheim and Kurtz 2018]

$$\min_{x \in X_d} f(x, d),$$

$$z_R = \min_{x \in \cap_{d \in U} X_d} \max_{d \in U} \frac{f(x, d) - f(x_d^*, d)}{f(x_d^*, d)}$$

Best in worst case, with cost=relative deviation from optimal

Chosen model of uncertainty

Different definitions of worst case based approaches

Absolute robust decision

Robust deviation decision

Relative robust decision

Chosen model of uncertainty

Different definitions of worst case based approaches

Absolute robust decision

Robust deviation decision

Relative robust decision

Why absolute robust decision?

- Avoid to compute optimal solutions of all scenarios

Chosen model of uncertainty

Different definitions of worst case based approaches

Absolute robust decision

Robust deviation decision

Relative robust decision

Uncertainty sets

Discrete set

Chosen model of uncertainty

Different definitions of worst case based approaches

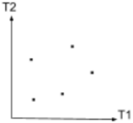
Absolute robust decision

Robust deviation decision

Relative robust decision

Uncertainty sets

Discrete set



[Li *et al.* 2011]

$U = \{c_1, \ldots, c_n\}$

- Good model
- Combinatorial

$\Longrightarrow$ explosion

Chosen model of uncertainty

Different definitions of worst case based approaches

Absolute robust decision

Robust deviation decision

Relative robust decision

Uncertainty sets

Discrete set

Interval set

Chosen model of uncertainty

Different definitions of worst case based approaches

Absolute robust decision

Robust deviation decision

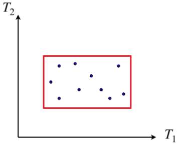
Relative robust decision

Uncertainty sets

Discrete set

Interval set

[Li *et al.* 2011]
$U = \Pi_{i=1}^{m}[a_i, b_i]$



- Do not consider correlation
- Easy

Chosen model of uncertainty

Different definitions of worst case based approaches

Absolute robust decision

Robust deviation decision

Relative robust decision

Uncertainty sets

Discrete set

Interval set

Ellipsoidal set

Chosen model of uncertainty

Different definitions of worst case based approaches

Absolute robust decision

Robust deviation decision

Relative robust decision

Uncertainty sets

Discrete set

Interval set

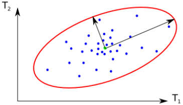
Ellipsoidal set

[Li *et al.* 2011]

Confidence region for multinormal distribution

$U = \{c \in \mathbb{R}^m; (c-\mu)^T \Sigma^{-1} (c-\mu) \leq \Omega^2\}$

- Consider correlation
- Hard



Chosen model of uncertainty

Different definitions of worst case based approaches

Absolute robust decision

Robust deviation decision

Relative robust decision

Uncertainty sets

Discrete set

Interval set

Ellipsoidal set

Chosen model of uncertainty

Different definitions of worst case based approaches

Absolute robust decision

Robust deviation decision

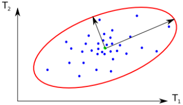
Relative robust decision

Uncertainty sets

Discrete set

Interval set

Ellipsoidal set



Why ellipsoidal uncertainty?

- Consider correlations
- Avoid the pessimism of general robust min-max optimization

Chosen model of uncertainty

Different definitions of worst case based approaches

Absolute robust decision

Robust deviation decision

Relative robust decision

Uncertainty sets

Discrete set

Interval set

Ellipsoidal set

Chosen problem to solve

General problems:

$$\min_{x \in X_d} f(x, d)$$

Chosen problem to solve

General problems:

$$\min_{x \in X_d} f(x, d)$$

Consider binary linear problems with uncertainty in cost

$$\min_{x \in X} c^T x, \quad X = \{x \in \{0, 1\}^m ; Ax = b\}$$

Chosen problem to solve

General problems:

$$\min_{x \in X_d} f(x, d)$$

Consider binary linear problems with uncertainty in cost

$$\min_{x \in X} c^T x, \quad X = \{x \in \{0, 1\}^m; Ax = b\}$$

Examples: Shortest path problem, k-median clustering problem, etc.

Complexity and existing methods

The absolute robust counterpart of binary linear problems under ellipsoidal uncertainty gives

$$\min_{x\in X}\max_{c\in U} c^T x \quad \rightarrow \min_{x\in X} \mu^T x + \sqrt{x^T\Sigma x} = \min_{x\in X} g(x) \qquad (1)$$

[Ilyina 2017] ($\mu^T x + \Omega\sqrt{x^T\Sigma x} \rightarrow \mu^T x + \sqrt{x^T\Sigma x}$: replace Σ by $\Omega^2\Sigma$)

Problem (1) is a binary non-linear problem $\implies$ NP-hard

Complexity and existing methods

The absolute robust counterpart of binary linear problems under ellipsoidal uncertainty gives

$$\min_{x \in X} \max_{c \in U} c^T x \quad \rightarrow \min_{x \in X} \mu^T x + \sqrt{x^T \Sigma x} = \min_{x \in X} g(x) \tag{1}$$

[Ilyina 2017] ($\mu^T x + \Omega\sqrt{x^T\Sigma x} \rightarrow \mu^T x + \sqrt{x^T \Sigma x}$: replace Σ by $\Omega^2\Sigma$)

Problem (1) is a binary non-linear problem $\Longrightarrow$ NP-hard

Methods to solve Problem (1)

Exact methods

Complexity and existing methods

The absolute robust counterpart of binary linear problems under ellipsoidal uncertainty gives

$$\min_{x\in X}\max_{c\in U} c^T x \quad \to \min_{x\in X} \mu^T x + \sqrt{x^T\Sigma x} = \min_{x\in X} g(x) \qquad (1)$$

[Ilyina 2017] ($\mu^T x + \Omega\sqrt{x^T\Sigma x} \to \mu^T x + \sqrt{x^T\Sigma x}$: replace Σ by $\Omega^2\Sigma$)

Problem (1) is a binary non-linear problem $\Longrightarrow$ NP-hard

Methods to solve Problem (1)

Exact methods

- Writing (1) as a Binary Second Order Cone Program permits us to use existing exact solvers based on Branch-and-Bound methods (e.g., CPLEX)
- Research work about exact methods are proposed :
- [Ilyina *et al* 2017]
- [Buchheim *et al.* 2017]: A Frank-Wolfe based Branch-and-bound algorithm

Complexity and existing methods

The absolute robust counterpart of binary linear problems under ellipsoidal uncertainty gives

$$\min_{x \in X} \max_{c \in U} c^T x \quad \rightarrow \min_{x \in X} \mu^T x + \sqrt{x^T \Sigma x} = \min_{x \in X} g(x) \qquad (1)$$

[Ilyina 2017] ($\mu^T x + \Omega\sqrt{x^T\Sigma x} \rightarrow \mu^T x + \sqrt{x^T\Sigma x}$: replace Σ by $\Omega^2\Sigma$)

Problem (1) is a binary non-linear problem $\Longrightarrow$ NP-hard

Methods to solve Problem (1)

Exact methods

Heuristic approaches

Complexity and existing methods

The absolute robust counterpart of binary linear problems under ellipsoidal uncertainty gives

$$\min_{x \in X} \max_{c \in U} c^T x \quad \rightarrow \min_{x \in X} \mu^T x + \sqrt{x^T \Sigma x} = \min_{x \in X} g(x) \tag{1}$$

[Ilyina 2017] ($\mu^T x + \Omega\sqrt{x^T \Sigma x} \rightarrow \mu^T x + \sqrt{x^T \Sigma x}$: replace Σ by $\Omega^2 \Sigma$)

Problem (1) is a binary non-linear problem $\implies$ NP-hard

Methods to solve Problem (1)

Exact methods

Heuristic approaches

There is no proposition of a heuristic approach

Exact methods are not scalable

$\implies$ Proposition of a heuristics with the idea of adapting Frank-Wolfe, this time for a heuristics

Outline

1. Context and positioning

2. A Frank-Wolfe based heuristic approach applied on the robust shortest path problem

3. Another Frank-Wolfe based heuristic approach applied on the robust k-median clustering problem

4. Conclusions and perspectives

Outline

1. Context and positioning

2. A Frank-Wolfe based heuristic approach applied on the robust shortest path problem

3. Another Frank-Wolfe based heuristic approach applied on the robust k-median clustering problem

4. Conclusions and perspectives

Postitioning

Recall the positioning

➡ Consider uncertainty in problems of the form

$$\min_{x \in X} c^T x, \quad X = \{x \in \{0,1\}^m; Ax = b\}$$

Postitioning

Recall the positioning

➡ Consider uncertainty in problems of the form

$$\min_{x \in X} c^T x, \quad X = \{x \in \{0,1\}^m; Ax = b\}$$

➡ The absolute robust decision under ellipsoidal uncertainty gives

$$\min_{x \in X} \max_{c \in U} c^T x \quad \rightarrow \quad \min_{x \in X} \mu^T x + \sqrt{x^T \Sigma x}$$

Postitioning

Recall the positioning

➡ Consider uncertainty in problems of the form

$$\min_{x \in X} c^T x, \quad X = \{x \in \{0,1\}^m; Ax = b\}$$

➡ The absolute robust decision under ellipsoidal uncertainty gives

$$\min_{x \in X} \max_{c \in U} c^T x \quad \rightarrow \quad \min_{x \in X} \mu^T x + \sqrt{x^T \Sigma x}$$

➡ The goal is to solve

$$\min_{x \in X} \mu^T x + \sqrt{x^T \Sigma x} = \min_{x \in X} g(x) \tag{2}$$

Postitioning

Recall the positioning

Consider uncertainty in problems of the form

$$\min_{x \in X} c^T x, \quad X = \{x \in \{0,1\}^m; Ax = b\}$$

The absolute robust decision under ellipsoidal uncertainty gives

$$\min_{x \in X} \max_{c \in U} c^T x \quad \rightarrow \quad \min_{x \in X} \mu^T x + \sqrt{x^T \Sigma x}$$

The goal is to solve

$$\min_{x \in X} \mu^T x + \sqrt{x^T \Sigma x} = \min_{x \in X} g(x) \tag{2}$$

Since, the work is based on Frank-Wolfe, recall first the classical Frank-Wolfe algorithm

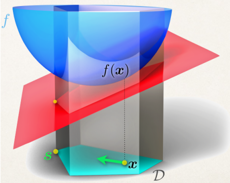
The classical Frank-Wolfe algorithm

f convex, continuously differentiable defined on a compact convex D.
Consider convex optimization problems of the form

$$\min_{x \in D} f(x)$$

Frank-Wolfe algorithm

```
Let $x^{(0)} \in D$
**for** $k = 0$ to $K$ **do**
    compute $s^{(k)} := \underset{s \in D}{\operatorname{argmin}} \nabla f(x^{(k)})^T s$
    update $x^{(k+1)} := (1 - \gamma^{(k)})x^{(k)} + \gamma^{(k)} s^{(k)}$
**end for**
```



[Jaggi 2013]

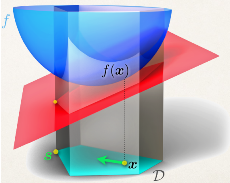
The classical Frank-Wolfe algorithm

f convex, continuously differentiable defined on a compact convex D.
Consider convex optimization problems of the form

$$\min_{x \in D} f(x)$$

Frank-Wolfe algorithm

```
Let $x^{(0)} \in D$
for $k = 0$ to $K$ do
    compute $s^{(k)} := \operatorname{argmin}_{s \in D} \nabla f(x^{(k)})^T s$
    update $x^{(k+1)} := (1 - \gamma^{(k)}) x^{(k)} + \gamma^{(k)} s^{(k)}$
end for
```



[Jaggi 2013]

$$s^{(k)} = \operatorname*{argmin}_{s \in D} [f(x^{(k)}) + \nabla f(x^{(k)})^T (s - x^{(k)})] = \operatorname*{argmin}_{s \in D} \nabla f(x^{(k)})^T s$$

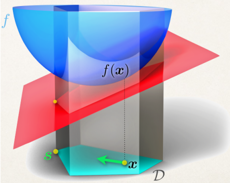
The classical Frank-Wolfe algorithm

f convex, continuously differentiable defined on a compact convex D.
Consider convex optimization problems of the form

$$\min_{x \in D} f(x)$$

Frank-Wolfe algorithm

```
Let $x^{(0)} \in D$
for $k = 0$ to $K$ do
    compute $s^{(k)} := \operatorname{argmin}_{s \in D} \nabla f(x^{(k)})^T s$
    update $x^{(k+1)} := (1 - \gamma^{(k)}) x^{(k)} + \gamma^{(k)} s^{(k)}$
end for
```



[Jaggi 2013]

Step-size

- $\gamma^{(k)} = \frac{2}{k+2}$
- $\gamma^{(k)} = \operatorname{argmin}_{\alpha \in [0,1]} f\big((1 - \alpha) x^{(k)} + \alpha s^{(k)}\big)$ (line search step)

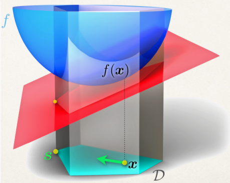
The classical Frank-Wolfe algorithm

f convex, continuously differentiable defined on a compact convex D.
Consider convex optimization problems of the form

$$\min_{x \in D} f(x)$$

Frank-Wolfe algorithm

```
Let $x^{(0)} \in D$
for $k = 0$ to $K$ do
    compute $s^{(k)} := \underset{s \in D}{\operatorname{argmin}} \nabla f(x^{(k)})^T s$
    update $x^{(k+1)} := (1 - \gamma^{(k)}) x^{(k)} + \gamma^{(k)} s^{(k)}$
end for
```



[Jaggi 2013]

Limits: Assumptions of Frank-Wolfe are not valid in our case

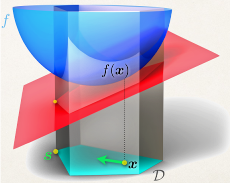
The classical Frank-Wolfe algorithm

f convex, continuously differentiable defined on a compact convex D. Consider convex optimization problems of the form

$$\min_{x \in D} f(x)$$

Frank-Wolfe algorithm

```
Let $x^{(0)} \in D$
for $k = 0$ to $K$ do
    compute $s^{(k)} := \underset{s \in D}{\operatorname{argmin}} \nabla f(x^{(k)})^T s$
    update $x^{(k+1)} := (1 - \gamma^{(k)})x^{(k)} + \gamma^{(k)} s^{(k)}$
end for
```



[Jaggi 2013]

- Limits: Assumptions of Frank-Wolfe are not valid in our case
- Need for an adapted algorithm

The proposed approach

The idea behind

➡ Use classical Frank-Wolfe to solve

$$\min_{x \in \textcolor{red}{\mathrm{Conv}(X)}} \mu^T x + \sqrt{x^T \Sigma x}$$

$X = \{x \in \{0,1\}^m ; Ax = b\}$
$\mathrm{Conv}(X) = \{x \in [0,1]^m ; Ax = b\}$

The proposed approach

The idea behind

➡ Use classical Frank-Wolfe to solve

$$\min_{x \in \text{Conv}(X)} \mu^T x + \sqrt{x^T \Sigma x}$$

$X = \{x \in \{0,1\}^m ; Ax = b\}$
$\text{Conv}(X) = \{x \in [0,1]^m ; Ax = b\}$

➡ The solution is not feasible

The proposed approach

The idea behind

➡ Use classical Frank-Wolfe to solve

$$\min_{x \in \mathrm{Conv}(X)} \mu^T x + \sqrt{x^T \Sigma x}$$

$X = \{x \in \{0,1\}^m ; Ax = b\}$
$\mathrm{Conv}(X) = \{x \in [0,1]^m ; Ax = b\}$

➡ The solution is not feasible

➡ The gradient is not well defined in zero

The proposed approach

The idea behind

➡ Use classical Frank-Wolfe to solve

$$\min_{x \in \mathrm{Conv}(X)} \mu^T x + \sqrt{x^T \Sigma x}$$

$X = \{x \in \{0,1\}^m; Ax = b\}$
$\mathrm{Conv}(X) = \{x \in [0,1]^m; Ax = b\}$

➡ The solution is not feasible

➡ The gradient is not well defined in zero

Before proceeding, some assumptions are needed

The proposed approach

First some assumptions

Three assumptions are necessary to our approach

(A1) For any real-valued vector a, there exists an efficient algorithm to solve $\min_{x \in X} a^T x$

(A2) For any real-valued vector a, there exists a solution for $\min_{x \in \mathrm{Conv(X)}} a^T x$ that belongs to X

(A3) $0_{\mathbb{R}^m}$ does not belong to X

Recall X, Conv(X)

$$X = \{x \in \{0,1\}^m ; Ax = b\}$$

$$\mathrm{Conv}(X) = \{x \in [0,1]^m ; Ax = b\}$$

The proposed approach

First some assumptions

Three assumptions are necessary to our approach

(A1) For any real-valued vector a, there exists an efficient algorithm to solve $\min_{x \in X} a^T x$

(A2) For any real-valued vector a, there exists a solution for $\min_{x \in \mathrm{Conv(X)}} a^T x$ that belongs to X

(A3) $0_{\mathbb{R}^m}$ does not belong to X

Recall X, Conv(X)

$$X = \{x \in \{0,1\}^m ; Ax = b\}$$

$$\mathrm{Conv}(X) = \{x \in [0,1]^m ; Ax = b\}$$

Thanks to **(A3)**, for all $x \in X$, the gradient of g is well defined

$$\nabla g(x) = \mu + \frac{\Sigma x}{\sqrt{x^T \Sigma x}}$$

The idea behind

➡ Use classical Frank-Wolfe to solve

$$\min_{x \in \mathrm{Conv}(X)} \mu^T x + \sqrt{x^T \Sigma x}$$

The idea behind

➡ Use classical Frank-Wolfe to solve

$$\min_{x \in \mathrm{Conv}(X)} \mu^T x + \sqrt{x^T \Sigma x}$$

➡ Consider the best intermediate solution, that is feasible thanks to the assumptions.

The proposed approach

DFW: a Frank-Wolfe based algorithm to solve (2)

```
$x^{(0)}$ a random feasible solution, $\varepsilon > 0$ close to zero, $K$ a maximum number of iterations.
$k \leftarrow 1$
stop $\leftarrow$ false
**while** $k \leq K$ and $\neg$stop **do**
    **if** $g(x^{(k-1)}) - g(x^{(k)}) < \varepsilon$: **then**
        stop $\leftarrow$ true
    **else**
        $s^{(k)} \in \underset{y \in \mathrm{Conv}(X)}{\mathrm{argmin}} \nabla g(x^{(k)})^T y$, with $s^{(k)} \in X$
        $\gamma^{(k)} \leftarrow \underset{\alpha \in [0,1]}{\mathrm{argmin}}\, g(x^{(k)} + \alpha(s^{(k)} - x^{(k)}))$
        $x^{(k+1)} \leftarrow (1 - \gamma^{(k)})x^{(k)} + \gamma^{(k)} s^{(k)}$
    **end if**
    $k++$
**end while**
return $\underset{s \in \{s^{(1)}, \ldots, s^{(k-1)}\}}{\mathrm{argmin}} g(s)$
```

The proposed approach

DFW: a Frank-Wolfe based algorithm to solve (2)

```
$x^{(0)}$ a random feasible solution, $\varepsilon > 0$ close to zero, $K$ a maximum number of iterations.
$k \leftarrow 1$
stop $\leftarrow$ false
**while** $k \leq K$ and $\neg$stop **do**
    **if** $g(x^{(k-1)}) - g(x^{(k)}) < \varepsilon$: **then**
        stop $\leftarrow$ true
    **else**
        $s^{(k)} \in \underset{y \in \mathrm{Conv}(X)}{\mathrm{argmin}} \; \nabla g(x^{(k)})^T y$, with $s^{(k)} \in X$
        $\gamma^{(k)} \leftarrow \underset{\alpha \in [0,1]}{\mathrm{argmin}} \, g(x^{(k)} + \alpha(s^{(k)} - x^{(k)}))$
        $x^{(k+1)} \leftarrow (1 - \gamma^{(k)})x^{(k)} + \gamma^{(k)} s^{(k)}$
    **end if**
    $k++$
**end while**
return $\underset{s \in \{s^{(1)}, \ldots, s^{(k-1)}\}}{\mathrm{argmin}} \; g(s)$
```

- $s^{(k)}$ minimum linear approximation
- $s^{(k)} \in X$ thanks to Assumption **(A2)**

The proposed approach

DFW: a Frank-Wolfe based algorithm to solve (2)

```
$x^{(0)}$ a random feasible solution, $\varepsilon > 0$ close to zero, $K$ a maximum number of iterations.
$k \leftarrow 1$
stop $\leftarrow$ false
**while** $k \leq K$ and $\neg$stop **do**
    **if** $g(x^{(k-1)}) - g(x^{(k)}) < \varepsilon$: **then**
        stop $\leftarrow$ true
    **else**
        $s^{(k)} \in \underset{y \in \mathrm{Conv}(X)}{\operatorname{argmin}} \nabla g(x^{(k)})^T y$, with $s^{(k)} \in X$
        $\gamma^{(k)} \leftarrow \underset{\alpha \in [0,1]}{\operatorname{argmin}} g(x^{(k)} + \alpha(s^{(k)} - x^{(k)}))$
        $x^{(k+1)} \leftarrow (1 - \gamma^{(k)})x^{(k)} + \gamma^{(k)} s^{(k)}$
    **end if**
    $k{+}{+}$
**end while**
return $\underset{s \in \{s^{(1)}, \ldots, s^{(k-1)}\}}{\operatorname{argmin}} g(s)$
```

- $s^{(k)}$ minimum linear approximation
- $s^{(k)} \in X$ thanks to Assumption **(A2)**

- Line search step: minimize $g(x^{(k+1)})$

The proposed approach

DFW: a Frank-Wolfe based algorithm to solve (2)

```
$x^{(0)}$ a random feasible solution, $\varepsilon > 0$ close to zero, $K$ a maximum number of iterations.
$k \leftarrow 1$
stop $\leftarrow$ false
**while** $k \leq K$ and $\neg$stop **do**
    **if** $g(x^{(k-1)}) - g(x^{(k)}) < \varepsilon$: **then**
        stop $\leftarrow$ true
    **else**
        $s^{(k)} \in \underset{y \in \mathrm{Conv}(X)}{\mathrm{argmin}} \nabla g(x^{(k)})^T y$, with $s^{(k)} \in X$
        $\gamma^{(k)} \leftarrow \underset{\alpha \in [0,1]}{\mathrm{argmin}}\, g(x^{(k)} + \alpha(s^{(k)} - x^{(k)}))$
        $x^{(k+1)} \leftarrow (1 - \gamma^{(k)})x^{(k)} + \gamma^{(k)} s^{(k)}$
    **end if**
    $k++$
**end while**
return $\underset{s \in \{s^{(1)}, \ldots, s^{(k-1)}\}}{\mathrm{argmin}} g(s)$
```

- $s^{(k)}$ minimum linear approximation
- $s^{(k)} \in X$ thanks to Assumption **(A2)**

- Line search step: minimize $g(x^{(k+1)})$

- $x^{(k)}$ converges in Conv(X)

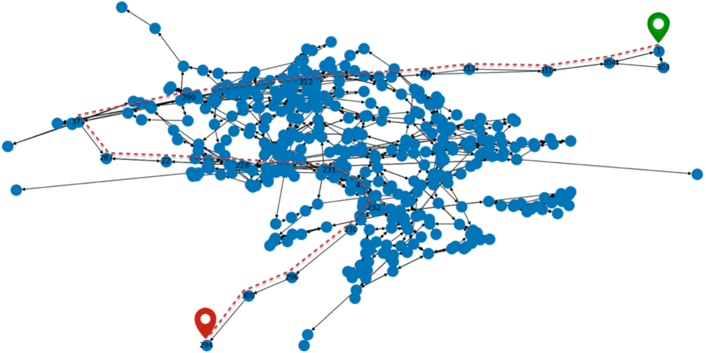
The proposed approach

DFW: a Frank-Wolfe based algorithm to solve (2)

```
$x^{(0)}$ a random feasible solution, $\varepsilon > 0$ close to zero, $K$
a maximum number of iterations.
$k \leftarrow 1$
stop $\leftarrow$ false
while $k \leq K$ and $\neg$stop do
    if $g(x^{(k-1)}) - g(x^{(k)}) < \varepsilon$: then
        stop $\leftarrow$ true
    else
        $s^{(k)} \in \underset{y \in \text{Conv}(X)}{\text{argmin}} \nabla g(x^{(k)})^T y$, with $s^{(k)} \in X$
        $\gamma^{(k)} \leftarrow \underset{\alpha \in [0,1]}{\text{argmin}}\, g(x^{(k)} + \alpha(s^{(k)} - x^{(k)}))$
        $x^{(k+1)} \leftarrow (1 - \gamma^{(k)})x^{(k)} + \gamma^{(k)} s^{(k)}$
    end if
    $k++$
end while
return $\underset{s \in \{s^{(1)}, \ldots, s^{(k-1)}\}}{\text{argmin}} g(s)$
```

- $s^{(k)}$ minimum linear approximation
- $s^{(k)} \in X$ thanks to Assumption **(A2)**
- Line search step: minimize $g(x^{(k+1)})$
- $x^{(k)}$ converges in Conv(X)
- We look at $s^{(k)}$: return the best one of all the iterations

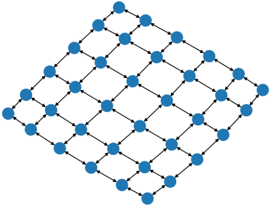
Numerical setup



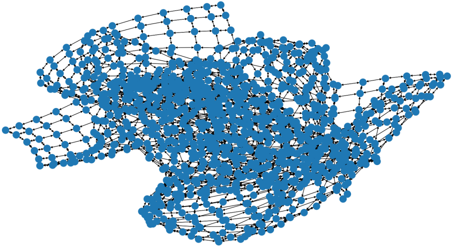
Berlin Mitte-Center graph with 398 nodes and 871 edges with an illustration of a path from node 1 to node 294

Numerical setup

Undirected grid graph $L \times L$, source node: 1, destination node L^2



Grid graph 6 × 6 with 36 nodes and 60 edges



Grid graph 34 × 34 with 1156 nodes and 2244 edges

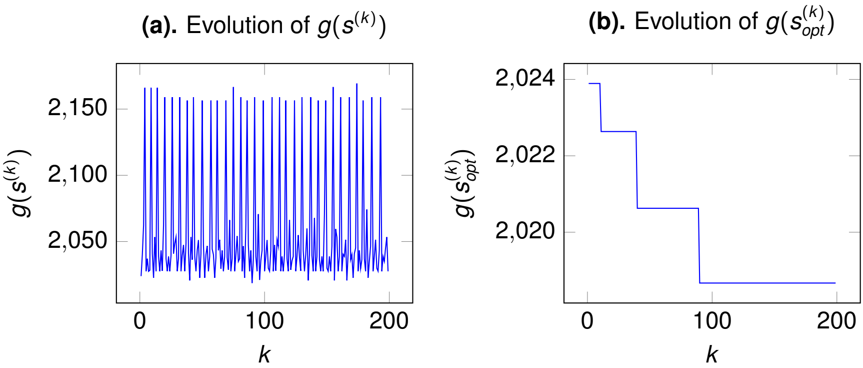
Numerical setup

Setup

- The robust shortest path problem from node 1 to node L^2
- $\Omega = 1$, (μ, Σ, x_0) random

Numerical results: behavior of the algorithm for $L = 34$

Denote, at each iteration k, the best solution so far as $s_{opt}^{(k)} = \text{argmin}_{s^{(l)} \in \{s^{(1)},\ldots,s^{(k)}\}} g(s^{(l)})$



Numerical results

Results

- Change size of the graph

Numerical results

Results

- Change size of the graph
- Compare with the solutions provided by `CPLEX`

Numerical results

Results

- Change size of the graph
- Compare with the solutions provided by `CPLEX`
- For small to medium graphs, DFW gives the same solution as CPLEX

Numerical results

Results

- Change size of the graph
- Compare with the solutions provided by `CPLEX`
- For small to medium graphs, DFW gives the same solution as CPLEX
- $L \geq 40$, branch-and-bound no more efficient

Numerical results

Results

- Change size of the graph
- Compare with the solutions provided by `CPLEX`
- For small to medium graphs, DFW gives the same solution as CPLEX
- $L \geq 40$, branch-and-bound no more efficient
- $L = 46$, `CPLEX` stops after 2 hours and a half

Numerical results

Results

- Change size of the graph
- Compare with the solutions provided by `CPLEX`
- For small to medium graphs, DFW gives the same solution as CPLEX
- $L \geq 40$, branch-and-bound no more efficient
- $L = 46$, `CPLEX` stops after 2 hours and a half

  200 iterations of DFW takes half an hour

Numerical results

Results

- Change size of the graph
- Compare with the solutions provided by `CPLEX`
- For small to medium graphs, DFW gives the same solution as CPLEX
- $L \geq 40$, branch-and-bound no more efficient
- $L = 46$, `CPLEX` stops after 2 hours and a half

  200 iterations of DFW takes half an hour

  Same solution as best integer of `CPLEX`

Outline

1. Context and positioning

2. A Frank-Wolfe based heuristic approach applied on the robust shortest path problem

3. Another Frank-Wolfe based heuristic approach applied on the robust $\mathtt{k}$-median clustering problem

4. Conclusions and perspectives

The $\mathtt{k}$-median clustering problem

Choose $\mathtt{k}$ clusters to minimize the sum of the distances between the points and their cluster centers

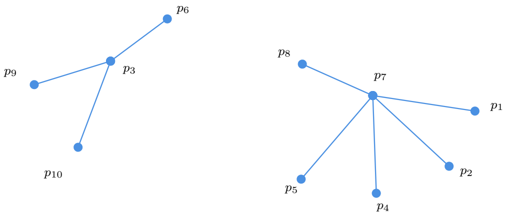
$$\min_{z \in \mathbb{R}^{n \times n}} \sum_{i=1}^{n} \sum_{j=1}^{n} d(p_i, p_j) z_{ij}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} z_{ij} = 1 \quad \forall j \in \{1, \dots, n\}$$

$$z_{ij} \leq y_i \quad \forall i, j \in \{1, \dots, n\}^2$$

$$\sum_{i=1}^{n} y_i = \mathtt{k}$$

$$z_{ij}, y_i \in \{0, 1\}$$



Simple example of a two cluster solution of a $\mathtt{k}$-median problem for $n = 10$ points

The $\mathtt{k}$-median clustering problem

Choose $\mathtt{k}$ clusters to minimize the sum of the distances between the points and their cluster centers

$$
\begin{aligned}
\min_{z \in \mathbb{R}^{n \times n}} \sum_{i=1}^{n} \sum_{j=1}^{n} d(p_i, p_j) z_{ij} & \\
\text{s.t. } \sum_{i=1}^{n} z_{ij} = 1 \quad \forall j \in \{1, \ldots, n\} & \\
z_{ij} \leq y_i \quad \forall i, j \in \{1, \ldots, n\}^2 & \\
\sum_{i=1}^{n} y_i = \mathtt{k} & \\
{\color{red} z_{ij}, y_i \in \{0, 1\}} &
\end{aligned}
$$

In a matrix representation,

$z_{ii} = y_i$, $i \in \{1, \ldots, n\}$

$y_i = 1 \implies p_i$ center

$z_{ij} = 1 \implies p_j$ associated to center p_i

The $\mathtt{k}$-median clustering problem

Choose $\mathtt{k}$ clusters to minimize the sum of the distances between the points and their cluster centers

$$\begin{aligned}
\min_{z \in \mathbb{R}^{n \times n}} \sum_{i=1}^{n} \sum_{j=1}^{n} d(p_i, p_j) z_{ij} \\
\text{s.t.} \quad \sum_{i=1}^{n} z_{ij} = 1 \quad \forall j \in \{1, \dots, n\} \\
z_{ij} \leq y_i \quad \forall i, j \in \{1, \dots, n\}^2 \\
\sum_{i=1}^{n} y_i = \mathtt{k} \\
z_{ij}, y_i \in \{0, 1\}
\end{aligned}$$

Constraint **(C1)**:

Sum of each column $= 1$

Every point is associated to one and only cluster center

The k-median clustering problem

Choose k clusters to minimize the sum of the distances between the points and their cluster centers

$$\min_{z \in \mathbb{R}^{n \times n}} \sum_{i=1}^{n} \sum_{j=1}^{n} d(p_i, p_j) z_{ij}$$

$$\text{s.t.} \sum_{i=1}^{n} z_{ij} = 1 \quad \forall j \in \{1, \dots, n\}$$

$$z_{ij} \leq y_i \quad \forall i, j \in \{1, \dots, n\}^2$$

$$\sum_{i=1}^{n} y_i = \mathrm{k}$$

$$z_{ij}, y_i \in \{0, 1\}$$

Constraint **(C2)**:

Non-diagonal $\leq$ diagonal

If associate point to point

$\implies$ the second is a center

The $\mathtt{k}$-median clustering problem

Choose $\mathtt{k}$ clusters to minimize the sum of the distances between the points and their cluster centers

$$\min_{z \in \mathbb{R}^{n \times n}} \sum_{i=1}^{n} \sum_{j=1}^{n} d(p_i, p_j) z_{ij}$$

$$\text{s.t.} \sum_{i=1}^{n} z_{ij} = 1 \quad \forall j \in \{1, \ldots, n\}$$

$$z_{ij} \leq y_i \quad \forall i, j \in \{1, \ldots, n\}^2$$

$$\sum_{i=1}^{n} y_i = \mathtt{k}$$

$$z_{ij}, y_i \in \{0, 1\}$$

Constraint **(C3)**:

Trace $= \mathtt{k}$

Exactly $\mathtt{k}$ centers

Presence of uncertainty

➡ If the distances are uncertain in the $\mathtt{k}$-median clustering problem

Presence of uncertainty

- If the distances are uncertain in the $\mathtt{k}$-median clustering problem
- Need for a robust solution

Presence of uncertainty

- If the distances are uncertain in the k-median clustering problem
- Need for a robust solution
- The k-median clustering problem can be written as

$$
\begin{aligned}
& \min d^T z \\
\text{s.t. } & \Sigma_{i=1}^{n} z_{n(i-1)+j} = 1 \quad \forall j \in \{1, \dots, n\} \\
& z_{n(i-1)+j} \leq z_{n(i-1)+i} \quad \forall i, j \in \{1, \dots, n\}^2 \\
& \Sigma_{i=1}^{n} z_{n(i-1)+i} = \mathtt{k} \\
& z \in \{0, 1\}^{n^2}
\end{aligned}
\tag{3}
$$

Presence of uncertainty

- If the distances are uncertain in the $\mathtt{k}$-median clustering problem
- Need for a robust solution
- The $\mathtt{k}$-median clustering problem can be written as

$$\begin{aligned} \min d^T z \\ \text{s.t. } \Sigma_{i=1}^{n} z_{n(i-1)+j} = 1 \quad \forall j \in \{1,\ldots,n\} \\ z_{n(i-1)+j} \leq z_{n(i-1)+i} \quad \forall i,j \in \{1,\ldots,n\}^2 \\ \Sigma_{i=1}^{n} z_{n(i-1)+i} = \mathtt{k} \\ z \in \{0,1\}^{n^2} \end{aligned} \tag{3}$$

- A flattening step
- A proof for equivalence of the definition of uncertainty between the two writings

Robust decision under ellipsoidal uncertainty

➡ The absolute robust $\mathtt{k}$-median clustering problem under ellipsoidal uncertainty is

$$\min_{z\in X} \mu^T z + \sqrt{z^T \Sigma z} \tag{4}$$

with

$$\begin{aligned} X = \Big\{ z \in \{0,1\}^{n^2} \text{ s.t.} \\ \Sigma_{i=1}^{n} z_{n(i-1)+j} = 1 \quad \forall j \in \{1,\dots,n\}, \\ z_{n(i-1)+j} \leq z_{n(i-1)+i} \forall i,j \in \{1,\dots,n\}^2, \\ \Sigma_{i=1}^{n} z_{n(i-1)+i} = \mathtt{k} \Big\} \end{aligned}$$

Robust decision under ellipsoidal uncertainty

The absolute robust $\mathtt{k}$-median clustering problem under ellipsoidal uncertainty is

$$\min_{z \in X} \mu^T z + \sqrt{z^T \Sigma z} \tag{4}$$

with

$$\begin{aligned} X = \Big\{ z \in \{0,1\}^{n^2} \text{ s.t.} \\ \Sigma_{i=1}^{n} z_{n(i-1)+j} = 1 \quad \forall j \in \{1,\dots,n\}, \\ z_{n(i-1)+j} \leq z_{n(i-1)+i} \forall i,j \in \{1,\dots,n\}^2, \\ \Sigma_{i=1}^{n} z_{n(i-1)+i} = \mathtt{k} \Big\} \end{aligned}$$

This follows the formulation of our study

Robust decision under ellipsoidal uncertainty

- The absolute robust k-median clustering problem under ellipsoidal uncertainty is

$$\min_{z \in X} \mu^T z + \sqrt{z^T \Sigma z} \qquad (4)$$

with

$$\begin{aligned} X = \Big\{ z \in \{0,1\}^{n^2} \text{ s.t.} \\ \Sigma_{i=1}^{n} z_{n(i-1)+j} = 1 \quad \forall j \in \{1,\dots,n\}, \\ z_{n(i-1)+j} \leq z_{n(i-1)+i} \forall i,j \in \{1,\dots,n\}^2, \\ \Sigma_{i=1}^{n} z_{n(i-1)+i} = \mathrm{k} \Big\} \end{aligned}$$

- This follows the formulation of our study
- Assumption **(A2)** not satisfied: no exact binarity relaxation
  $\Longrightarrow$ cannot apply DFW

Robust decision under ellipsoidal uncertainty

The absolute robust $\mathtt{k}$-median clustering problem under ellipsoidal uncertainty is

$$\min_{z \in X} \mu^T z + \sqrt{z^T \Sigma z} \tag{4}$$

with

$$\begin{aligned} X = \Big\{ z \in \{0,1\}^{n^2} \text{ s.t.} \\ \Sigma_{i=1}^{n} z_{n(i-1)+j} = 1 \quad \forall j \in \{1,\ldots,n\}, \\ z_{n(i-1)+j} \leq z_{n(i-1)+i} \forall i,j \in \{1,\ldots,n\}^2, \\ \Sigma_{i=1}^{n} z_{n(i-1)+i} = \mathtt{k} \Big\} \end{aligned}$$

This follows the formulation of our study

Assumption **(A2)** not satisfied: no exact binarity relaxation
$\Longrightarrow$ cannot apply DFW

We propose another Frank-Wolfe based algorithm for the robust $\mathtt{k}$-median clustering

The proposed approach

The idea behind

➡ Use classical Frank-Wolfe to solve

$$\min_{x \in \text{Conv}(X)} \mu^T x + \sqrt{x^T \Sigma x}$$

The proposed approach

The idea behind

Use classical Frank-Wolfe to solve

$$\min_{x \in \mathrm{Conv}(X)} \mu^T x + \sqrt{x^T \Sigma x}$$

$$\begin{aligned} \mathrm{Conv}(X) = \Big\{ z \in [0,1]^{n^2} \text{ s.t.} \\ \Sigma_{i=1}^{n} z_{n(i-1)+j} = 1 \quad \forall j \in \{1,\dots,n\}, \\ z_{n(i-1)+j} \leq z_{n(i-1)+i} \forall i,j \in \{1,\dots,n\}^2, \\ \Sigma_{i=1}^{n} z_{n(i-1)+i} = \mathrm{k} \Big\} \end{aligned}$$

Conv(X): Constraints **(C1)**, **(C2)**, **(C3)** satisfied, variables not binary

The proposed approach

The idea behind

⇨ Use classical Frank-Wolfe to solve

$$\min_{x \in \mathrm{Conv}(X)} \mu^T x + \sqrt{x^T \Sigma x}$$

$$\begin{aligned} \mathrm{Conv}(X) = \Big\{ z \in [0,1]^{n^2} \text{ s.t.} \\ \Sigma_{i=1}^{n} z_{n(i-1)+j} = 1 \quad \forall j \in \{1,\ldots,n\}, \\ z_{n(i-1)+j} \leq z_{n(i-1)+i} \forall i,j \in \{1,\ldots,n\}^2, \\ \Sigma_{i=1}^{n} z_{n(i-1)+i} = \mathrm{k} \Big\} \end{aligned}$$

Conv(X): Constraints **(C1)**, **(C2)**, **(C3)** satisfied, variables not binary

⇨ Consider the feasible round of the mean of all the intermediate solutions.

The proposed approach

MFW: a Frank-Wolfe based algorithm to solve Problem (4)

```
$x^{(0)} \in \text{Conv}(X)$ a random solution, $\varepsilon > 0$ close to zero,
$\mathring{K}$ a maximum number of iterations.
$\mathring{k} \leftarrow 1$
stop $\leftarrow$ false
**while** $\mathring{k} \leq \mathring{K}$ and $\neg$stop **do**
    **if** $g(x^{(\mathring{k}-1)}) - g(x^{(\mathring{k})}) < \varepsilon$: **then**
        stop $\leftarrow$ true
    **else**
        $s^{(\mathring{k})} \in \underset{y \in \text{Conv}(X)}{\text{argmin}} \nabla g(x^{(\mathring{k})})^T y$
        $\gamma^{(\mathring{k})} \leftarrow \underset{\alpha \in [0,1]}{\text{argmin}}\, g(x^{(\mathring{k})} + \alpha(s^{(\mathring{k})} - x^{(\mathring{k})}))$
        $x^{(\mathring{k}+1)} \leftarrow (1 - \gamma^{(\mathring{k})})x^{(\mathring{k})} + \gamma^{(\mathring{k})} s^{(\mathring{k})}$
    **end if**
    $\mathring{k}++$
**end while**
return a feasible round of $\mu_{\mathring{k}-1} = \frac{\sum_{i=1}^{\mathring{k}-1} s^{(i)}}{\mathring{k} - 1}$
```

The proposed approach

MFW: a Frank-Wolfe based algorithm to solve Problem (4)

```
$x^{(0)} \in \mathrm{Conv}(X)$ a random solution, $\varepsilon > 0$ close to zero,
  $\mathring{K}$ a maximum number of iterations.
$\mathring{k} \leftarrow 1$
stop $\leftarrow$ false
while $\mathring{k} \leq \mathring{K}$ and $\neg$stop do
    if $g(x^{(\mathring{k}-1)}) - g(x^{(\mathring{k})}) < \varepsilon$: then
        stop $\leftarrow$ true
    else
        $s^{(\mathring{k})} \in \underset{y \in \mathrm{Conv}(X)}{\mathrm{argmin}} \; \nabla g(x^{(\mathring{k})})^T y$
        $\gamma^{(\mathring{k})} \leftarrow \underset{\alpha \in [0,1]}{\mathrm{argmin}} \, g(x^{(\mathring{k})} + \alpha(s^{(\mathring{k})} - x^{(\mathring{k})}))$
        $x^{(\mathring{k}+1)} \leftarrow (1 - \gamma^{(\mathring{k})})x^{(\mathring{k})} + \gamma^{(\mathring{k})} s^{(\mathring{k})}$
    end if
    $\mathring{k}++$
end while
return a feasible round of $\mu_{\mathring{k}-1} = \frac{\Sigma_{i=1}^{\mathring{k}-1} s^{(i)}}{\mathring{k}-1}$
```

- $s^{(k)}$ minimum linear approximation
- $s^{(k)} \notin X \implies$ need to round

The proposed approach

MFW: a Frank-Wolfe based algorithm to solve Problem (4)

```
$x^{(0)} \in \mathrm{Conv}(X)$ a random solution, $\varepsilon > 0$ close to zero,
$\mathring{K}$ a maximum number of iterations.
$\mathring{k} \leftarrow 1$
stop $\leftarrow$ false
**while** $\mathring{k} \leq \mathring{K}$ and $\neg$stop **do**
    **if** $g(x^{(\mathring{k}-1)}) - g(x^{(\mathring{k})}) < \varepsilon$: **then**
        stop $\leftarrow$ true
    **else**
        $s^{(\mathring{k})} \in \underset{y \in \mathrm{Conv}(X)}{\mathrm{argmin}}\ \nabla g(x^{(\mathring{k})})^T y$
        $\gamma^{(\mathring{k})} \leftarrow \underset{\alpha \in [0,1]}{\mathrm{argmin}}\, g(x^{(\mathring{k})} + \alpha(s^{(\mathring{k})} - x^{(\mathring{k})}))$
        $x^{(\mathring{k}+1)} \leftarrow (1 - \gamma^{(\mathring{k})}) x^{(\mathring{k})} + \gamma^{(\mathring{k})} s^{(\mathring{k})}$
    **end if**
    $\mathring{k}++$
**end while**
return a feasible round of $\mu_{\mathring{k}-1} = \dfrac{\Sigma_{i=1}^{\mathring{k}-1} s^{(i)}}{\mathring{k}-1}$
```

- $s^{(k)}$ minimum linear approximation
- $s^{(k)} \notin X \implies$ need to round

- Line search step: minimize $g(x^{(k+1)})$

The proposed approach

MFW: a Frank-Wolfe based algorithm to solve Problem (4)

```
$x^{(0)} \in \mathrm{Conv}(X)$ a random solution, $\varepsilon > 0$ close to zero,
  $\mathring{K}$ a maximum number of iterations.
$\mathring{k} \leftarrow 1$
stop $\leftarrow$ false
**while** $\mathring{k} \leq \mathring{K}$ and $\neg$stop **do**
    **if** $g(x^{(\mathring{k}-1)}) - g(x^{(\mathring{k})}) < \varepsilon$: **then**
        stop $\leftarrow$ true
    **else**
        $s^{(\mathring{k})} \in \operatorname*{argmin}_{y \in \mathrm{Conv}(X)} \nabla g(x^{(\mathring{k})})^T y$
        $\gamma^{(\mathring{k})} \leftarrow \operatorname*{argmin}_{\alpha \in [0,1]} g(x^{(\mathring{k})} + \alpha(s^{(\mathring{k})} - x^{(\mathring{k})}))$
        $x^{(\mathring{k}+1)} \leftarrow (1 - \gamma^{(\mathring{k})})x^{(\mathring{k})} + \gamma^{(\mathring{k})} s^{(\mathring{k})}$
    **end if**
    $\mathring{k} + +$
**end while**
return a feasible round of $\mu_{\mathring{k}-1} = \frac{\Sigma_{i=1}^{\mathring{k}-1} s^{(i)}}{\mathring{k} - 1}$
```

- $s^{(k)}$ minimum linear approximation
- $s^{(k)} \notin X \implies$ need to round
- Line search step: minimize $g(x^{(k+1)})$
- $x^{(k)}$ converges in $\mathrm{Conv}(X)$

The proposed approach

MFW: a Frank-Wolfe based algorithm to solve Problem (4)

```
$x^{(0)} \in \mathrm{Conv}(X)$ a random solution, $\varepsilon > 0$ close to zero,
$\mathring{K}$ a maximum number of iterations.
$\mathring{k} \leftarrow 1$
stop $\leftarrow$ false
while $\mathring{k} \leq \mathring{K}$ and $\neg$stop do
    if $g(x^{(\mathring{k}-1)}) - g(x^{(\mathring{k})}) < \varepsilon$: then
        stop $\leftarrow$ true
    else
        $s^{(\mathring{k})} \in \underset{y \in \mathrm{Conv}(X)}{\mathrm{argmin}} \nabla g(x^{(\mathring{k})})^T y$
        $\gamma^{(\mathring{k})} \leftarrow \underset{\alpha \in [0,1]}{\mathrm{argmin}}\, g(x^{(\mathring{k})} + \alpha(s^{(\mathring{k})} - x^{(\mathring{k})}))$
        $x^{(\mathring{k}+1)} \leftarrow (1 - \gamma^{(\mathring{k})})x^{(\mathring{k})} + \gamma^{(\mathring{k})} s^{(\mathring{k})}$
    end if
    $\mathring{k}++$
end while
return a feasible round of $\mu_{\mathring{k}-1} = \frac{\sum_{i=1}^{\mathring{k}-1} s^{(i)}}{\mathring{k}-1}$
```

- $s^{(k)}$ minimum linear approximation
- $s^{(k)} \notin X \implies$ need to round
- Line search step: minimize $g(x^{(k+1)})$
- $x^{(k)}$ converges in $\mathrm{Conv}(X)$
- We look at $s^{(k)}$: return the feasible round of the mean of all the iterations

A feasible rounding algorithm: example of a 2-median clustering with 10 points

$$\begin{bmatrix}
0.53 & 0 & 0.53 & 0 & 0 & 0 & 0 & 0.05 & 0 & 0.53 \\
0 & 0.14 & 0.14 & 0.14 & 0.02 & 0 & 0.14 & 0.14 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.39 & 0 & 0.38 & 0.38 & 0.38 & 0 & 0 & 0.32 & 0.25 \\
0 & 0 & 0 & 0.09 & 0.09 & 0.09 & 0.09 & 0 & 0 & 0.09 \\
0 & 0 & 0 & 0.04 & 0.04 & 0.04 & 0 & 0 & 0.04 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.47 & 0.47 & 0.01 & 0 & 0.46 & 0.14 & 0.43 & 0.47 & 0.3 & 0.13 \\
0 & 0 & 0.32 & 0.34 & 0 & 0.34 & 0.34 & 0.34 & 0.34 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

A feasible rounding algorithm: example of a 2-median clustering with 10 points

$$
\begin{bmatrix}
{\color{blue}0.53} & 0 & 0.53 & 0 & 0 & 0 & 0 & 0.05 & 0 & 0.53 \\
0 & 0.14 & 0.14 & 0.14 & 0.02 & 0 & 0.14 & 0.14 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.39 & 0 & 0.38 & 0.38 & 0.38 & 0 & 0 & 0.32 & 0.25 \\
0 & 0 & 0 & 0.09 & 0.09 & 0.09 & 0.09 & 0 & 0 & 0.09 \\
0 & 0 & 0 & 0.04 & 0.04 & 0.04 & 0 & 0 & 0.04 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.47 & 0.47 & 0.01 & 0 & 0.46 & 0.14 & 0.43 & {\color{blue}0.47} & 0.3 & 0.13 \\
0 & 0 & 0.32 & 0.34 & 0 & 0.34 & 0.34 & 0.34 & 0.34 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\rightarrow
\begin{bmatrix}
{\color{blue}1} & & & & & & & 0 & & \\
0 & 0 & & & & & & 0 & & \\
0 & & 0 & & & & & 0 & & \\
0 & & & 0 & & & & 0 & & \\
0 & & & & 0 & & & 0 & & \\
0 & & & & & 0 & & 0 & & \\
0 & & & & & & 0 & 0 & & \\
0 & & & & & & & {\color{blue}1} & & \\
0 & & & & & & & 0 & 0 & \\
0 & & & & & & & 0 & & 0
\end{bmatrix}
$$

Sort the diagonal elements, and choose the 2 biggest elements

A feasible rounding algorithm: example of a 2-median clustering with 10 points

$$\begin{bmatrix} 0.53 & 0 & 0.53 & 0 & 0 & 0 & 0 & 0.05 & 0 & 0.53 \\ 0 & 0.14 & 0.14 & 0.14 & 0.02 & 0 & 0.14 & 0.14 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.39 & 0 & 0.38 & 0.38 & 0.38 & 0 & 0 & 0.32 & 0.25 \\ 0 & 0 & 0 & 0.09 & 0.09 & 0.09 & 0.09 & 0 & 0 & 0.09 \\ 0 & 0 & 0 & 0.04 & 0.04 & 0.04 & 0 & 0 & 0.04 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.47 & 0.47 & 0.01 & 0 & 0.46 & 0.14 & 0.43 & 0.47 & 0.3 & 0.13 \\ 0 & 0 & 0.32 & 0.34 & 0 & 0.34 & 0.34 & 0.34 & 0.34 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & & & & & & 0 & & \\ 0 & & 0 & & & & & 0 & & \\ 0 & & & 0 & & & & 0 & & \\ 0 & & & & 0 & & & 0 & & \\ 0 & & & & & 0 & & 0 & & \\ 0 & & & & & & 0 & 0 & & \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & & & & & & & 0 & 0 & \\ 0 & & & & & & & 0 & & 0 \end{bmatrix}$$

Sort the diagonal elements, and choose the 2 biggest elements

In reduced matrix, sort each column

A feasible rounding algorithm: example of a 2-median clustering with 10 points

$$\begin{bmatrix} 0.53 & 0 & 0.53 & 0 & 0 & 0 & 0 & 0.05 & 0 & 0.53 \\ 0 & 0.14 & 0.14 & 0.14 & 0.02 & 0 & 0.14 & 0.14 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.39 & 0 & 0.38 & 0.38 & 0.38 & 0 & 0 & 0.32 & 0.25 \\ 0 & 0 & 0 & 0.09 & 0.09 & 0.09 & 0.09 & 0 & 0 & 0.09 \\ 0 & 0 & 0 & 0.04 & 0.04 & 0.04 & 0 & 0 & 0.04 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.47 & 0.47 & 0.01 & 0 & 0.46 & 0.14 & 0.43 & 0.47 & 0.3 & 0.13 \\ 0 & 0 & 0.32 & 0.34 & 0 & 0.34 & 0.34 & 0.34 & 0.34 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Sort the diagonal elements, and choose the 2 biggest elements

In reduced matrix, sort each column

Then the rest equals zero, and the rounding is done!

Numerical setup

Setup

- We changed the number of points *n* for the $\mathtt{k}$-median problem for $\mathtt{k} = 2$
- We compared with the solutions provided by `CPLEX`
- $\Omega = 1$, and for every *n*, 80 different (μ, Σ, x_0)

Numerical results of MFW Algorithm

$$E_r = \frac{g(\hat{x}) - p^*}{p^*} \qquad \#\{E_r = 0\}$$

n	Time(s) of CPLEX	Time(s) MFW	$\#\{E_r = 0\}$	$\overline{E_r}$
5	0.1644	5.0149	55 %	0.0555
6	0.5424	7.8	71.25 %	0.0513
7	0.8296	12.9796	48.75 %	0.0486
8	0.9948	6.9707	67.5 %	0.0186
9	1.9202	9.6168	63.75 %	0.0246
10	2.1028	16.6282	58.75 %	0.0432
11	2.1607	14.1045	70 %	0.0447
12	3.6378	19.778	23 %	0.0862
13	4.1873	17.8977	35 %	0.0694

Numerical results of MFW Algorithm

$$E_r = \frac{g(\hat{x}) - p^*}{p^*} \qquad \#\{E_r = 0\}$$

n	Time(s) of CPLEX	Time(s) MFW	$\#\{E_r = 0\}$	$\overline{E_r}$
5	0.1644	5.0149	55 %	0.0555
6	0.5424	7.8	71.25 %	0.0513
7	0.8296	12.9796	48.75 %	0.0486
8	0.9948	6.9707	67.5 %	0.0186
9	1.9202	9.6168	63.75 %	0.0246
10	2.1028	16.6282	58.75 %	0.0432
11	2.1607	14.1045	70 %	0.0447
12	3.6378	19.778	23 %	0.0862
13	4.1873	17.8977	35 %	0.0694

➡ The relative error is in average small (0.0186 to 0.0862)

Numerical results of MFW Algorithm

$$E_r = \frac{g(\hat{x}) - p^*}{p^*} \qquad \#\{E_r = 0\}$$

n	Time(s) of CPLEX	Time(s) MFW	$\#\{E_r = 0\}$	$\overline{E_r}$
5	0.1644	5.0149	55 %	0.0555
6	0.5424	7.8	71.25 %	0.0513
7	0.8296	12.9796	48.75 %	0.0486
8	0.9948	6.9707	67.5 %	0.0186
9	1.9202	9.6168	63.75 %	0.0246
10	2.1028	16.6282	58.75 %	0.0432
11	2.1607	14.1045	70 %	0.0447
12	3.6378	19.778	23 %	0.0862
13	4.1873	17.8977	35 %	0.0694

➡ The relative error equals zero in up to 70% of the cases

Numerical results of MFW Algorithm

$$E_r = \frac{g(\hat{x}) - p^*}{p^*} \qquad \#\{E_r = 0\}$$

n	Time(s) of CPLEX	Time(s) MFW	$\#\{E_r = 0\}$	$\overline{E_r}$
5	0.1644	5.0149	55 %	0.0555
6	0.5424	7.8	71.25 %	0.0513
7	0.8296	12.9796	48.75 %	0.0486
8	0.9948	6.9707	67.5 %	0.0186
9	1.9202	9.6168	63.75 %	0.0246
10	2.1028	16.6282	58.75 %	0.0432
11	2.1607	14.1045	70 %	0.0447
12	3.6378	19.778	23 %	0.0862
13	4.1873	17.8977	35 %	0.0694

CPLEX is faster but the difference in time between MFW Algorithm and CPLEX is not very big (around $1/5$ in average)

Outline

1. Context and positioning

2. A Frank-Wolfe based heuristic approach applied on the robust shortest path problem

3. Another Frank-Wolfe based heuristic approach applied on the robust `k`-median clustering problem

4. Conclusions and perspectives

Conclusion

- Considering the uncertainty in optimization problems is important

Conclusion

- Considering the uncertainty in optimization problems is important
- The robust problem with an ellipsoidal uncertainty of a binary linear problem in the cost function is a hard problem

Conclusion

- Considering the uncertainty in optimization problems is important
- The robust problem with an ellipsoidal uncertainty of a binary linear problem in the cost function is a hard problem
- We propose a heuristic approach based on Frank-Wolfe applied on the robust shortest path problem

Conclusion

- Considering the uncertainty in optimization problems is important
- The robust problem with an ellipsoidal uncertainty of a binary linear problem in the cost function is a hard problem
- We propose a heuristic approach based on Frank-Wolfe applied on the robust shortest path problem
- Numerical results show that the heuristic approach gives the optimal solution

Conclusion

- Considering the uncertainty in optimization problems is important
- The robust problem with an ellipsoidal uncertainty of a binary linear problem in the cost function is a hard problem
- We propose a heuristic approach based on Frank-Wolfe applied on the robust shortest path problem
- Numerical results show that the heuristic approach gives the optimal solution
- An extension on the robust $\mathtt{k}$-median clustering has been studied

Conclusion

- Considering the uncertainty in optimization problems is important
- The robust problem with an ellipsoidal uncertainty of a binary linear problem in the cost function is a hard problem
- We propose a heuristic approach based on Frank-Wolfe applied on the robust shortest path problem
- Numerical results show that the heuristic approach gives the optimal solution
- An extension on the robust k-median clustering has been studied
- A heuristic algorithm based on Frank-Wolfe has been proposed, with a feasible rounding algorithm

Conclusion

- Considering the uncertainty in optimization problems is important
- The robust problem with an ellipsoidal uncertainty of a binary linear problem in the cost function is a hard problem
- We propose a heuristic approach based on Frank-Wolfe applied on the robust shortest path problem
- Numerical results show that the heuristic approach gives the optimal solution
- An extension on the robust $\mathtt{k}$-median clustering has been studied
- A heuristic algorithm based on Frank-Wolfe has been proposed, with a feasible rounding algorithm
- Results show that this algorithm gives the optimal solution in most of the cases, and that it gives close-to-optimal solutions when they are not optimal

Perspectives

- DFW Algorithm could be improved (better results, less processing time- for example using the away step FW)

Perspectives

- DFW Algorithm could be improved (better results, less processing time- for example using the away step FW)
- More numerical results could include a study of the algorithm's parameters, and more tests with larger instances

Perspectives

- DFW Algorithm could be improved (better results, less processing time-for example using the away step FW)
- More numerical results could include a study of the algorithm's parameters, and more tests with larger instances
- DFW is a heuristics $\Longrightarrow$ it has certainly some limitations $\Longrightarrow$ a distance from optimality, in its worst case?

Perspectives

- DFW Algorithm could be improved (better results, less processing time- for example using the away step FW)
- More numerical results could include a study of the algorithm's parameters, and more tests with larger instances
- DFW is a heuristics $\implies$ it has certainly some limitations $\implies$ a distance from optimality, in its worst case?
- Improve MFW Algorithm (e.g., by improving the rounding technique)

Perspectives

- DFW Algorithm could be improved (better results, less processing time- for example using the away step FW)
- More numerical results could include a study of the algorithm's parameters, and more tests with larger instances
- DFW is a heuristics $\Longrightarrow$ it has certainly some limitations $\Longrightarrow$ a distance from optimality, in its worst case?
- Improve MFW Algorithm (e.g., by improving the rounding technique)
- Test MFW with different uncertainty configurations of the clusters

Thank you for your attention