

Convex integer optimization with Frank-Wolfe methods

Sebastian Pokutta

joint work with: Mathieu Besançon, Deborah Hendrych, and Hannah Troppens

Technische Universität Berlin
and
Zuse Institute Berlin

pokutta@math.tu-berlin.de
@spokutta

6th RIKEN-IMI-ISM-NUS-ZIB-MODAL-NHR Workshop on
Advances in Classical and Quantum Algorithms for Optimization
and Machine Learning

September 19th, 2022 · Tokyo, Japan



Berlin Mathematics Research Center



What is this talk about?

Introduction

*A mixed-integer convex optimization method
based on conditional gradients.*

What is this talk about?

Introduction

*A mixed-integer convex optimization method
based on conditional gradients.*

Why? Conditional gradients generate *sparse* iterates, leading to lower fractionality, and hence less branching.

What is this talk about?

Introduction

*A mixed-integer convex optimization method
based on conditional gradients.*

Why? Conditional gradients generate *sparse* iterates, leading to lower fractionality, and hence less branching.

Today: A brief overview of approach and solver.

What is this talk about?

Introduction

A mixed-integer convex optimization method based on conditional gradients.

Why? Conditional gradients generate *sparse* iterates, leading to lower fractionality, and hence less branching.

Today: A brief overview of approach and solver.

Outline

- Recap: Conditional Gradients a.k.a. the Frank-Wolfe algorithm
- Mixed-Integer Conditional Gradients
- Julia Package Boscia.jl

(Hyperlinked) References are not exhaustive; check references contained therein.



Conditional Gradients a.k.a. the Frank-Wolfe algorithm

—The Basics—

The basic problem

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Given a smooth and convex function f and a polytope P , solve **optimization problem**:

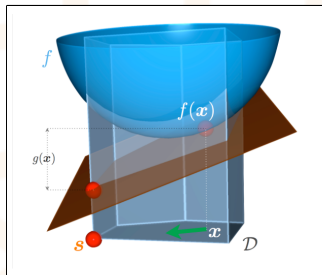
The basic problem

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Given a smooth and convex function f and a polytope P , solve **optimization problem**:

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

(baseProblem)



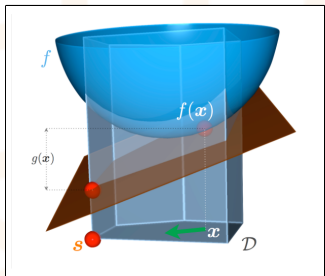
Source: [Jaggi, 2013]

The basic problem

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Given a smooth and convex function f and a polytope P , solve **optimization problem**:

$$\min_{x \in P} f(x) \quad (\text{baseProblem})$$



Source: [Jaggi, 2013]

1. Very **versatile** model
2. Can use various types of **information about both f and P**
3. Works very well in (continuous) **real-world applications**
4. At the core of many (all?) **learning algorithms** (albeit mostly non-convex case)

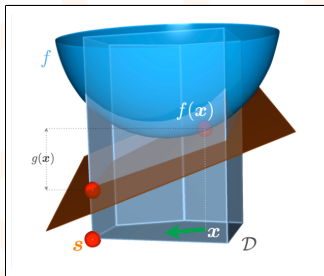
The basic problem

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Given a smooth and convex function f and a polytope P , solve **optimization problem**:

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

(baseProblem)



Source: [Jaggi, 2013]

Our setup.

1. Access to P . **Linear Minimization Oracle (LMO)**: Given linear objective c return

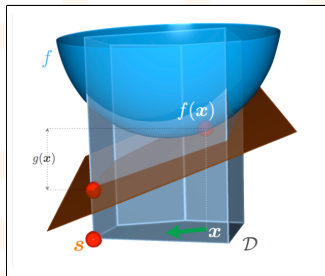
$$x \leftarrow \arg \min_{v \in P} c^T v.$$

The basic problem

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Given a smooth and convex function f and a polytope P , solve **optimization problem**:

$$\min_{x \in P} f(x) \quad (\text{baseProblem})$$



Source: [Jaggi, 2013]

Our setup.

1. Access to P . **Linear Minimization Oracle (LMO)**: Given linear objective c return

$$x \leftarrow \arg \min_{v \in P} c^T v.$$

2. Access to f . **First-Order Oracle (FO)**: Given x return

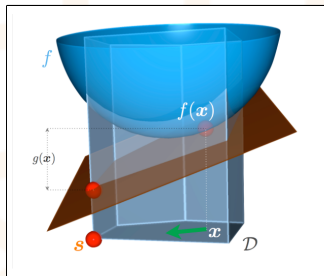
$$\nabla f(x) \quad \text{and} \quad f(x).$$

The basic problem

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Given a smooth and convex function f and a polytope P , solve **optimization problem**:

$$\min_{x \in P} f(x) \quad (\text{baseProblem})$$



Source: [Jaggi, 2013]

Our setup.

1. Access to P . **Linear Minimization Oracle (LMO)**: Given linear objective c return

$$x \leftarrow \arg \min_{v \in P} c^T v.$$

2. Access to f . **First-Order Oracle (FO)**: Given x return

$$\nabla f(x) \quad \text{and} \quad f(x).$$

\Rightarrow *Complexity of convex optimization relative to LO/FO oracle*

Interlude: why LMOs?

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

LMO model has many advantages.

1. Includes explicit formulation via constraints
2. Some problems do not possess 'small' formulations but have efficient LMOs.
Example: Matching Polytope [Rothvoss, 2014, Braun and Pokutta, 2015a,b, Braun et al., 2015, 2017a]
3. Allows modeling of compact convex constraints as long as we have an LMO.
Example: SDP cone
4. Often much faster than projection.
Example: nuclear norm. Largest singular vector (Lanczos method) vs. full SVD
5. LMO is a black box for the algorithms
6. For many LMOs of interest close form solutions available.
Example: ℓ_1 -ball for LASSO regression.

For an overview see: [Combettes and Pokutta, 2021]

The basic problem

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Basic notions. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a differentiable function.

The basic problem

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Basic notions. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a differentiable function.

Definition (Convexity)

For all x, y it holds:

$$f(y) - f(x) \geq \langle \nabla f(x), y - x \rangle.$$

In particular, all local minima are global minima.

The basic problem

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Basic notions. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a differentiable function.

Definition (Convexity)

For all x, y it holds:

$$f(y) - f(x) \geq \langle \nabla f(x), y - x \rangle.$$

In particular, all local minima are global minima.

Definition (L -Smoothness)

For all x, y it holds:

$$f(y) - f(x) \leq \langle \nabla f(x), y - x \rangle + \frac{L}{2} \|y - x\|^2.$$

The basic problem

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Basic notions. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a differentiable function.

Definition (Convexity)

For all x, y it holds:

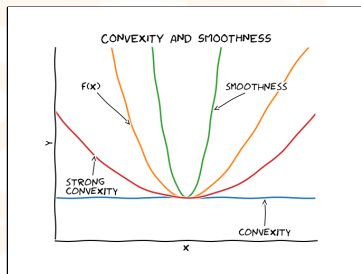
$$f(y) - f(x) \geq \langle \nabla f(x), y - x \rangle.$$

In particular, all local minima are global minima.

Definition (L -Smoothness)

For all x, y it holds:

$$f(y) - f(x) \leq \langle \nabla f(x), y - x \rangle + \frac{L}{2} \|y - x\|^2.$$

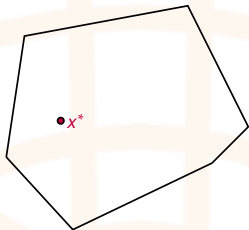


The Frank-Wolfe Algorithm

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Algorithm Frank-Wolfe Algorithm (FW)

- 1: $x_0 \in P$
 - 2: **for** $t = 0$ **to** $T - 1$ **do**
 - 3: $v_t \leftarrow \arg \min_{v \in P} \langle \nabla f(x_t), v \rangle$
 - 4: $x_{t+1} \leftarrow x_t + \gamma_t(v_t - x_t)$
 - 5: **end for**
-



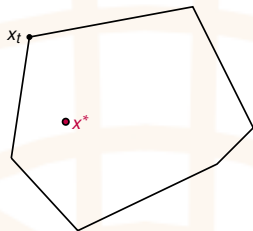
[Frank and Wolfe, 1956, Levitin and Polyak, 1966]

The Frank-Wolfe Algorithm

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Algorithm Frank-Wolfe Algorithm (FW)

- 1: $x_0 \in P$
 - 2: **for** $t = 0$ **to** $T - 1$ **do**
 - 3: $v_t \leftarrow \arg \min_{v \in P} \langle \nabla f(x_t), v \rangle$
 - 4: $x_{t+1} \leftarrow x_t + \gamma_t(v_t - x_t)$
 - 5: **end for**
-



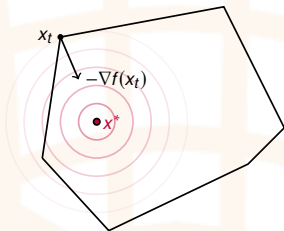
[Frank and Wolfe, 1956, Levitin and Polyak, 1966]

The Frank-Wolfe Algorithm

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Algorithm Frank-Wolfe Algorithm (FW)

- 1: $x_0 \in P$
 - 2: **for** $t = 0$ **to** $T - 1$ **do**
 - 3: $v_t \leftarrow \arg \min_{v \in P} \langle \nabla f(x_t), v \rangle$
 - 4: $x_{t+1} \leftarrow x_t + \gamma_t(v_t - x_t)$
 - 5: **end for**
-



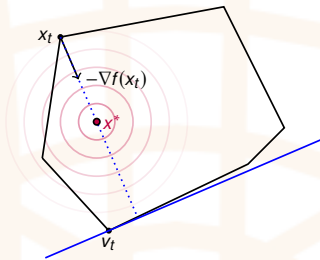
[Frank and Wolfe, 1956, Levitin and Polyak, 1966]

The Frank-Wolfe Algorithm

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Algorithm Frank-Wolfe Algorithm (FW)

- 1: $x_0 \in P$
 - 2: **for** $t = 0$ **to** $T - 1$ **do**
 - 3: $v_t \leftarrow \arg \min_{v \in P} \langle \nabla f(x_t), v \rangle$
 - 4: $x_{t+1} \leftarrow x_t + \gamma_t(v_t - x_t)$
 - 5: **end for**
-



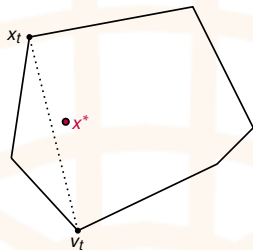
[Frank and Wolfe, 1956, Levitin and Polyak, 1966]

The Frank-Wolfe Algorithm

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Algorithm Frank-Wolfe Algorithm (FW)

- 1: $x_0 \in P$
 - 2: **for** $t = 0$ **to** $T - 1$ **do**
 - 3: $v_t \leftarrow \arg \min_{v \in P} \langle \nabla f(x_t), v \rangle$
 - 4: $x_{t+1} \leftarrow x_t + \gamma_t(v_t - x_t)$
 - 5: **end for**
-



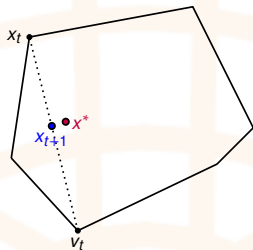
[Frank and Wolfe, 1956, Levitin and Polyak, 1966]

The Frank-Wolfe Algorithm

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Algorithm Frank-Wolfe Algorithm (FW)

- 1: $x_0 \in P$
 - 2: **for** $t = 0$ **to** $T - 1$ **do**
 - 3: $v_t \leftarrow \arg \min_{v \in P} \langle \nabla f(x_t), v \rangle$
 - 4: $x_{t+1} \leftarrow x_t + \gamma_t(v_t - x_t)$
 - 5: **end for**
-



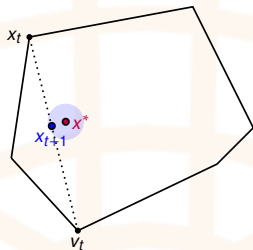
[Frank and Wolfe, 1956, Levitin and Polyak, 1966]

The Frank-Wolfe Algorithm

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Algorithm Frank-Wolfe Algorithm (FW)

- 1: $x_0 \in P$
 - 2: **for** $t = 0$ **to** $T - 1$ **do**
 - 3: $v_t \leftarrow \arg \min_{v \in P} \langle \nabla f(x_t), v \rangle$
 - 4: $x_{t+1} \leftarrow x_t + \gamma_t(v_t - x_t)$
 - 5: **end for**
-



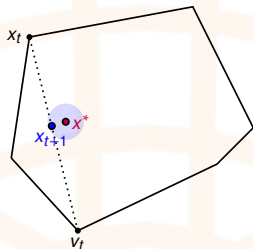
[Frank and Wolfe, 1956, Levitin and Polyak, 1966]

The Frank-Wolfe Algorithm

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Algorithm Frank-Wolfe Algorithm (FW)

- 1: $x_0 \in P$
 - 2: **for** $t = 0$ **to** $T - 1$ **do**
 - 3: $v_t \leftarrow \arg \min_{v \in P} \langle \nabla f(x_t), v \rangle$
 - 4: $x_{t+1} \leftarrow x_t + \gamma_t(v_t - x_t)$
 - 5: **end for**
-



[Frank and Wolfe, 1956, Levitin and Polyak, 1966]

Advantages:

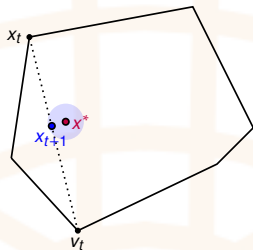
- **Extremely simple and robust:** no complicated data structures to maintain
- **Easy to implement:** requires only the two oracles
- **Projection-free:** feasibility convex combination and LO oracle.
- **Sparsity:** optimal solution is a convex combination of (usually) vertices.

The Frank-Wolfe Algorithm

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Algorithm Frank-Wolfe Algorithm (FW)

- 1: $x_0 \in P$
 - 2: **for** $t = 0$ **to** $T - 1$ **do**
 - 3: $v_t \leftarrow \arg \min_{v \in P} \langle \nabla f(x_t), v \rangle$
 - 4: $x_{t+1} \leftarrow x_t + \gamma_t(v_t - x_t)$
 - 5: **end for**
-



[Frank and Wolfe, 1956, Levitin and Polyak, 1966]

Advantages:

- **Extremely simple and robust:** no complicated data structures to maintain
- **Easy to implement:** requires only the two oracles
- **Projection-free:** feasibility convex combination and LO oracle.
- **Sparsity:** optimal solution is a convex combination of (usually) vertices.

Disadvantages:

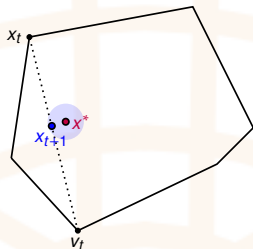
- Suboptimal convergence rate of $O(1/T)$

The Frank-Wolfe Algorithm

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Algorithm Frank-Wolfe Algorithm (FW)

- 1: $x_0 \in P$
 - 2: **for** $t = 0$ **to** $T - 1$ **do**
 - 3: $v_t \leftarrow \arg \min_{v \in P} \langle \nabla f(x_t), v \rangle$
 - 4: $x_{t+1} \leftarrow x_t + \gamma_t(v_t - x_t)$
 - 5: **end for**
-



[Frank and Wolfe, 1956, Levitin and Polyak, 1966]

Advantages:

- **Extremely simple and robust:** no complicated data structures to maintain
- **Easy to implement:** requires only the two oracles
- **Projection-free:** feasibility convex combination and LO oracle.
- **Sparsity:** optimal solution is a convex combination of (usually) vertices.

Disadvantages:

- Suboptimal convergence rate of $O(1/T)$

⇒ *Despite (theoretically) suboptimal rate heavily used in applications due to simplicity.*

Simple Convergence Proof

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Theorem (Convergence rate of the vanilla Frank-Wolfe Algorithm)

Let f be L -smooth convex, P be polytope with diameter D . With choice $\gamma_t \doteq \frac{2}{t+3}$:

$$f(x_t) - f(x^*) \leq \frac{2LD^2}{t+3}.$$

Simple Convergence Proof

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Theorem (Convergence rate of the vanilla Frank-Wolfe Algorithm)

Let f be L -smooth convex, P be polytope with diameter D . With choice $\gamma_t \doteq \frac{2}{t+3}$:

$$f(x_t) - f(x^*) \leq \frac{2LD^2}{t+3}.$$

Proof Sketch.

By smoothness:

$$f(x_{t+1}) - f(x_t) \leq \langle \nabla f(x_t), x_{t+1} - x_t \rangle + \frac{L}{2} \|x_{t+1} - x_t\|^2 = \gamma_t \langle \nabla f(x_t), v_t - x_t \rangle + \frac{L\gamma_t^2}{2} \|v_t - x_t\|^2.$$

Simple Convergence Proof

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Theorem (Convergence rate of the vanilla Frank-Wolfe Algorithm)

Let f be L -smooth convex, P be polytope with diameter D . With choice $\gamma_t \doteq \frac{2}{t+3}$:

$$f(x_t) - f(x^*) \leq \frac{2LD^2}{t+3}.$$

Proof Sketch.

By smoothness:

$$f(x_{t+1}) - f(x_t) \leq \langle \nabla f(x_t), x_{t+1} - x_t \rangle + \frac{L}{2} \|x_{t+1} - x_t\|^2 = \gamma_t \langle \nabla f(x_t), v_t - x_t \rangle + \frac{L\gamma_t^2}{2} \|v_t - x_t\|^2.$$

LP maximality and convexity: $\langle \nabla f(x_t), v_t - x_t \rangle \leq \langle \nabla f(x_t), x^* - x_t \rangle \leq f(x^*) - f(x_t)$. Moreover, $\|v_t - x_t\| \leq D$.

Simple Convergence Proof

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Theorem (Convergence rate of the vanilla Frank-Wolfe Algorithm)

Let f be L -smooth convex, P be polytope with diameter D . With choice $\gamma_t \doteq \frac{2}{t+3}$:

$$f(x_t) - f(x^*) \leq \frac{2LD^2}{t+3}.$$

Proof Sketch.

By smoothness:

$$f(x_{t+1}) - f(x_t) \leq \langle \nabla f(x_t), x_{t+1} - x_t \rangle + \frac{L}{2} \|x_{t+1} - x_t\|^2 = \gamma_t \langle \nabla f(x_t), v_t - x_t \rangle + \frac{L\gamma_t^2}{2} \|v_t - x_t\|^2.$$

LP maximality and convexity: $\langle \nabla f(x_t), v_t - x_t \rangle \leq \langle \nabla f(x_t), x^* - x_t \rangle \leq f(x^*) - f(x_t)$. Moreover, $\|v_t - x_t\| \leq D$.

Thus:

$$f(x_{t+1}) - f(x^*) \leq (1 - \gamma_t)(f(x_t) - f(x^*)) + \gamma_t^2 \frac{LD^2}{2}.$$

Simple Convergence Proof

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

Theorem (Convergence rate of the vanilla Frank-Wolfe Algorithm)

Let f be L -smooth convex, P be polytope with diameter D . With choice $\gamma_t \doteq \frac{2}{t+3}$:

$$f(x_t) - f(x^*) \leq \frac{2LD^2}{t+3}.$$

Proof Sketch.

By smoothness:

$$f(x_{t+1}) - f(x_t) \leq \langle \nabla f(x_t), x_{t+1} - x_t \rangle + \frac{L}{2} \|x_{t+1} - x_t\|^2 = \gamma_t \langle \nabla f(x_t), v_t - x_t \rangle + \frac{L\gamma_t^2}{2} \|v_t - x_t\|^2.$$

LP maximality and convexity: $\langle \nabla f(x_t), v_t - x_t \rangle \leq \langle \nabla f(x_t), x^* - x_t \rangle \leq f(x^*) - f(x_t)$. Moreover, $\|v_t - x_t\| \leq D$.

Thus:

$$f(x_{t+1}) - f(x^*) \leq (1 - \gamma_t)(f(x_t) - f(x^*)) + \gamma_t^2 \frac{LD^2}{2}.$$

By Induction (plugging in the guarantee + definition of γ_t):

$$f(x_{t+1}) - f(x^*) \leq \left(1 - \frac{2}{t+3}\right) \frac{2LD^2}{t+3} + \frac{4}{(t+3)^2} \cdot \frac{LD^2}{2} = \frac{2LD^2(t+2)}{(t+3)^2} \leq \frac{2LD^2}{t+4},$$

by $(t+2)(t+4) \leq (t+3)^2$.

Significant progress over the recent years (incomplete list)

Conditional Gradients a.k.a. the Frank-Wolfe algorithm

1. Strongly convex case [Garber and Hazan, 2013, Lacoste-Julien and Jaggi, 2015, Lan and Zhou, 2016, Garber and Meshi, 2016]
2. Non-convex case [Lacoste-Julien, 2016]
3. Online case [Hazan and Kale, 2012]
4. Stochastic variants and adaptive gradients [Hazan and Luo, 2016, Reddi et al., 2016, Combettes et al., 2020]
5. Sharp functions and sharp regions [Kerdreux et al., 2019, 2021a,b]
6. Acceleration [Diakonikolas et al., 2020, Bach, 2020, Carderera et al., 2021]
7. Specialized variants [Freund et al., 2017, Braun et al., 2017b, 2019b,a]

Conditional Gradients very competitive: simple, robust, real-world performance.

For more background etc see our survey!

[Braun et al., 2022]

Mixed-Integer Conditional Gradients

—The Framework—

Problem setting

Mixed-Integer Conditional Gradients

Basically. Smooth convex objective over MIPs.

Problem setting

Mixed-Integer Conditional Gradients

Basically. Smooth convex objective over MIPs.

Slightly more general:

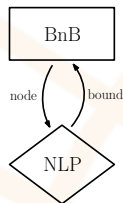
$$\begin{aligned} \min_{x,y} f(x,y) \\ \text{s.t. } x \in \mathcal{X} \\ x_j \in \mathbb{Z} \quad \forall j \in J \\ y \in \mathcal{Y} \end{aligned}$$

with LMO over $(\mathcal{X} \cap \text{bounds}) \times \mathcal{Y}$.

Three main algorithmic frameworks for MINLPs

Mixed-Integer Conditional Gradients

Diamond blocks represent nodal relaxations in the given framework.

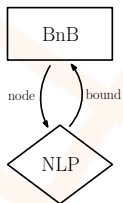


Standard BnB framework on top of NLP relaxations.

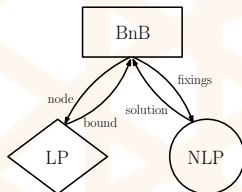
Three main algorithmic frameworks for MINLPs

Mixed-Integer Conditional Gradients

Diamond blocks represent nodal relaxations in the given framework.



Standard BnB framework on top of NLP relaxations.

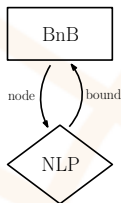


LP-based MINLP frameworks and outer approximations

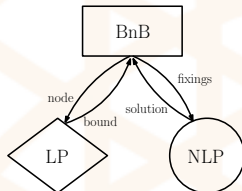
Three main algorithmic frameworks for MINLPs

Mixed-Integer Conditional Gradients

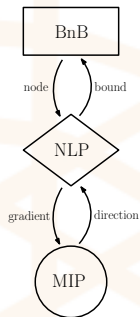
Diamond blocks represent nodal relaxations in the given framework.



Standard BnB framework on top of NLP relaxations.



LP-based MINLP frameworks and outer approximations

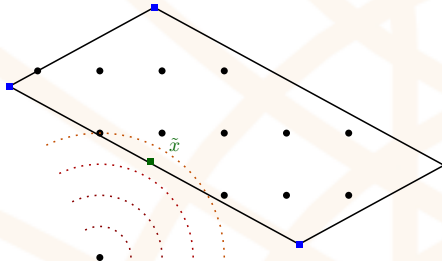


Our approach. Linearized models solved as MIPs within the Frank-Wolfe algorithm on top of which we branch

Tree of trees or forest → **Boscia (Corsican) = Forest.**

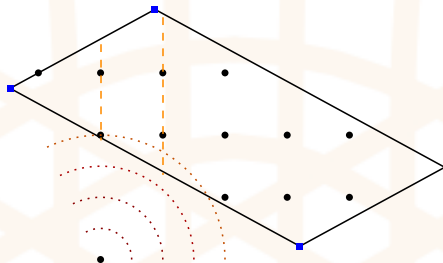
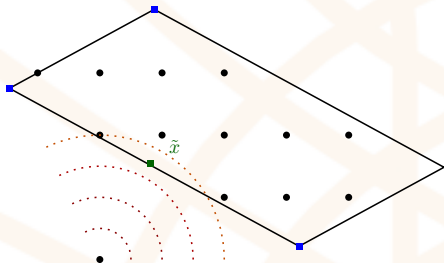
Branching: continuous relaxation (usual approach)

Mixed-Integer Conditional Gradients



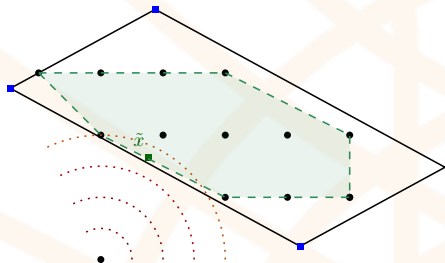
Branching: continuous relaxation (usual approach)

Mixed-Integer Conditional Gradients



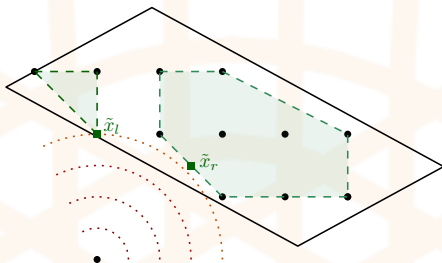
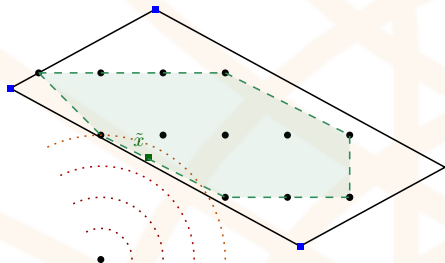
Branching: mixed-integer hull (our approach)

Mixed-Integer Conditional Gradients



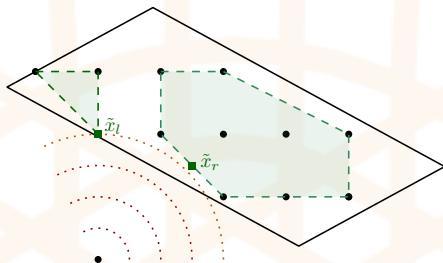
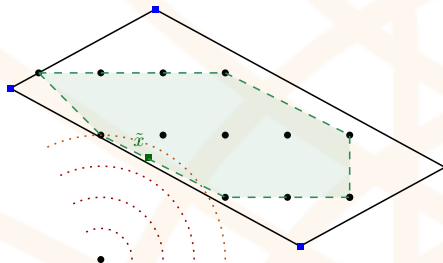
Branching: mixed-integer hull (our approach)

Mixed-Integer Conditional Gradients



Branching: mixed-integer hull (our approach)

Mixed-Integer Conditional Gradients



Open question.

Can we define adaptive criteria to choose relaxation?

(E.g., geometry of the feasible set, conditioning of the function)

Reducing number of MIP oracle calls

Mixed-Integer Conditional Gradients

We use **Blended Pairwise Conditional Gradients (BPCG)**

[Tsuji et al., 2022]

Reducing number of MIP oracle calls

Mixed-Integer Conditional Gradients

We use **Blended Pairwise Conditional Gradients (BPCG)**

[Tsuji et al., 2022]

- **Lazification.** aggressively reuse old solutions

Reducing number of MIP oracle calls

Mixed-Integer Conditional Gradients

We use **Blended Pairwise Conditional Gradients (BPCG)**

[Tsuji et al., 2022]

- **Lazification.** aggressively reuse old solutions
- **Blending.** perform local steps for sparsity → low fractionality

Reducing number of MIP oracle calls

Mixed-Integer Conditional Gradients

We use **Blended Pairwise Conditional Gradients (BPCG)**

[Tsuji et al., 2022]

- **Lazification.** aggressively reuse old solutions
- **Blending.** perform local steps for sparsity \rightarrow low fractionality
- **Active set.** branching means simply splitting convex combination

Reducing number of MIP oracle calls

Mixed-Integer Conditional Gradients

We use **Blended Pairwise Conditional Gradients (BPCG)**

[Tsuji et al., 2022]

- **Lazification.** aggressively reuse old solutions
- **Blending.** perform local steps for sparsity → low fractionality
- **Active set.** branching means simply splitting convex combination
- **Discarded set.** reuse solutions from previous nodes

Reducing number of MIP oracle calls

Mixed-Integer Conditional Gradients

We use **Blended Pairwise Conditional Gradients (BPCG)**

[Tsuji et al., 2022]

- **Lazification.** aggressively reuse old solutions
- **Blending.** perform local steps for sparsity → low fractionality
- **Active set.** branching means simply splitting convex combination
- **Discarded set.** reuse solutions from previous nodes
- **Incomplete resolution and warmstarts.** Less work per node

Does it help?

Reducing number of MIP oracle calls

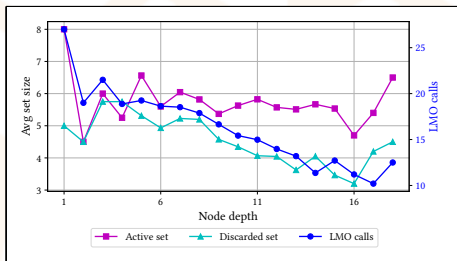
Mixed-Integer Conditional Gradients

We use **Blended Pairwise Conditional Gradients (BPCG)**

[Tsuji et al., 2022]

- **Lazification.** aggressively reuse old solutions
- **Blending.** perform local steps for sparsity → low fractionality
- **Active set.** branching means simply splitting convex combination
- **Discarded set.** reuse solutions from previous nodes
- **Incomplete resolution and warmstarts.** Less work per node

Does it help?



On average something like 7 to 10 sub-MIPs per node.

Reducing cost for each MIP

Mixed-Integer Conditional Gradients

Subproblems are MIPs. **Leverage MIP advances:**

- Cutting-planes
- Domain propagation
- Presolving
- Primal heuristics
- etc.

Reducing cost for each MIP

Mixed-Integer Conditional Gradients

Subproblems are MIPs. **Leverage MIP advances:**

- Cutting-planes
- Domain propagation
- Presolving
- Primal heuristics
- etc.

Moreover, we can **reuse information across solves** heavily:

- MIP solver called with different objectives within node
- Identical polyhedron with updated bounds solved across nodes
- All found primal solutions are valid for main problem

Reducing cost for each MIP

Mixed-Integer Conditional Gradients

Subproblems are MIPs. **Leverage MIP advances:**

- Cutting-planes
- Domain propagation
- Presolving
- Primal heuristics
- etc.

Moreover, we can **reuse information across solves** heavily:

- MIP solver called with different objectives within node
- Identical polyhedron with updated bounds solved across nodes
- All found primal solutions are valid for main problem

Question of MIP reoptimization:

[Gamrath et al., 2015]

Which information should be (conditionally) transferred across instances?



Boscia.jl

—The Code—

The package

Boscia.jl

- Julia package
- Based on `Bonobo.jl` (BnB package) and `FrankWolfe.jl` (our FW package)
- Via `MOI` can use basically any MIP solver; some features specific to `SCIP`
- Includes other features such as hybrid branching
- Available under MIT license

Example: Code

Boscia.jl

```
using Boscia
using FrankWolfe
using Random
using SCIP
using LinearAlgebra
import MathOptInterface
const MOI = MathOptInterface

n = 6

const diffw = 0.5 * ones(n)
o = SCIP.Optimizer()

MOI.set(o, MOI.Silent(), true)

x = MOI.add_variables(o, n)

for xi in x
    MOI.add_constraint(o, xi, MOI.GreaterThan(0.0))
    MOI.add_constraint(o, xi, MOI.LessThan(1.0))
    MOI.add_constraint(o, xi, MOI.ZeroOne())
end

lmo = FrankWolfe.MathOptLMO(o)

function f(x)
    return sum(0.5*(x.-diffw).^2)
end

function grad!(storage, x)
    @. storage = x-diffw
end

x, _, result = Boscia.solve(f, grad!, lmo, verbose = true)
```

Example: planted solution in high-dimensional space

Boscia.jl

```
julia> include("examples/low_dim_in_high_dim.jl")
```

Boscia Algorithm.

Parameter settings.

```
Tree traversal strategy: Move best bound  
Branching strategy: Most infeasible  
Absolute dual gap tolerance: 1.000000e-06  
Relative dual gap tolerance: 1.000000e-02  
Frank-Wolfe subproblem tolerance: 1.000000e-05  
Total number of variables: 12  
Number of integer variables: 0  
Number of binary variables: 12
```

Iteration	Open	Bound	Incumbent	Gap (abs)	Gap (rel)	Time (s)	Nodes/sec	FW (ms)	LMO (ms)	LMO (calls c)	FW (Its)	#ActiveSet	Discarded	
*	1	2	-2.297891e+03	-1.977988e+03	3.199322e+02	1.617490e-01	9.150000e-01	3.278689e+00	758	3	11	10001	1	0
*	2	3	-2.297891e+03	-2.238338e+03	5.955322e+01	2.660600e-02	1.442000e+00	3.467406e+00	526	11	46	10001	17	0
*	4	5	-2.292074e+03	-2.239976e+03	5.209869e+01	2.325860e-02	2.500000e+00	3.600000e+00	531	6	101	10001	11	0
*	5	6	-2.292074e+03	-2.242301e+03	4.977302e+01	2.219729e-02	3.024000e+00	3.637566e+00	523	10	133	10001	12	0
*	6	7	-2.292074e+03	-2.242301e+03	4.977302e+01	2.219729e-02	3.544000e+00	3.668172e+00	519	11	164	10001	4	0
*	16	17	-2.282246e+03	-2.243023e+03	3.922226e+01	1.748534e-02	8.726000e+00	3.781802e+00	524	8	439	10001	6	2
*	21	22	-2.280549e+03	-2.243325e+03	3.722381e+01	1.659314e-02	1.131100e+01	3.801609e+00	514	5	564	10001	7	1
*	29	30	-2.279719e+03	-2.244814e+03	3.490439e+01	1.534887e-02	1.544900e+01	3.819017e+00	522	8	761	10001	3	1
*	66	67	-2.271953e+03	-2.245231e+03	2.672273e+01	1.190200e-02	3.451800e+01	3.853062e+00	517	8	1616	10001	1	2
*	100	101	-2.268603e+03	-2.245231e+03	2.337210e+01	1.040967e-02	5.204200e+01	3.862265e+00	516	6	2357	10001	2	1
*	119	120	-2.267387e+03	-2.245231e+03	2.215595e+01	9.868008e-03	6.184900e+01	3.864250e+00	530	6	2778	10001	4	0

Solution Statistics.

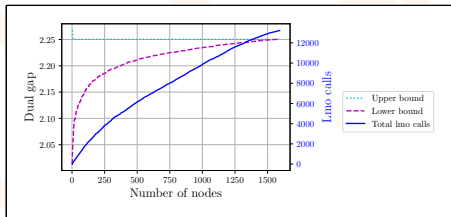
```
Solution Status: Optimal (tolerance reached)  
Primal Objective: -2245.23067557406  
Dual Bound: -2267.3866295644566  
Dual Gap (relative): 0.009868007876175864
```

Search Statistics.

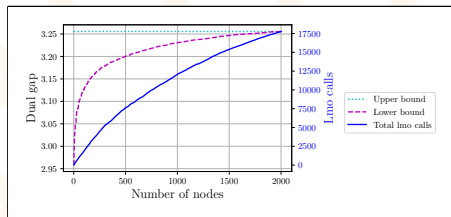
```
Total number of nodes processed: 239  
Total number of lmo calls: 2782  
Total time (s): 61.853  
LMO calls / sec: 44.9776082000873  
Nodes / sec: 3.8640001293389163  
LMO calls / node: 11.640167364016737
```

Example: computational results: Sparse Regression

Boscia.jl



High-dimensional sparse regression problem with ℓ_0 -constraints



High-dimensional sparse regression problem over mixed-integer feasible region.

Thank you!

Preprint: arxiv.org/abs/2208.11010,
Convex integer optimization with FW methods
Package available at github.com/ZIB-IOL/Boscia.jl

[Hendrych et al., 2022]

In a nutshell. Minimize smooth convex objective over any MIP. Applications in engineering, sparse prediction models, statistics, and relaxation of combinatorial problems.

References I

- F. Bach. On the effectiveness of Richardson extrapolation in machine learning. *arXiv preprint 2002.02835v3*, July 2020.
- G. Braun and S. Pokutta. The matching polytope does not admit fully-polynomial size relaxation schemes. *Proceedings of SODA*, 2015a.
- G. Braun and S. Pokutta. The matching polytope does not admit fully-polynomial size relaxation schemes. *IEEE Transactions on Information Theory*, 61(10):1–11, 2015b.
- G. Braun, S. Pokutta, and D. Zink. Inapproximability of combinatorial problems via small LPs and SDPs. *Proceedings of STOC*, 2015.
- G. Braun, R. Jain, T. Lee, and S. Pokutta. Information-theoretic approximations of the nonnegative rank. *Computational Complexity*, 26(1):147–197, 2017a.
- G. Braun, S. Pokutta, and D. Zink. Lazifying Conditional Gradient Algorithms. *Proceedings of the International Conference on Machine Learning (ICML)*, 2017b.
- G. Braun, S. Pokutta, D. Tu, and S. Wright. Blended Conditional Gradients: the unconditioning of conditional gradients. *Proceedings of ICML*, 2019a.
- G. Braun, S. Pokutta, and D. Zink. Lazifying Conditional Gradient Algorithms. *Journal of Machine Learning Research (JMLR)*, 20(71):1–42, 2019b.
- G. Braun, A. Carderera, C. W. Combettes, H. Hassani, A. Karbasi, A. Mokthari, and S. Pokutta. Conditional gradient methods. *preprint*, 8 2022.
- A. Carderera, J. Diakonikolas, C. Y. Lin, and S. Pokutta. Parameter-free Locally Accelerated Conditional Gradients. *to appear in Proceedings of ICML*, 2 2021.
- C. W. Combettes and S. Pokutta. Complexity of Linear Minimization and Projection on Some Sets. *to appear in Operations Research Letters*, 1 2021.
- C. W. Combettes, C. Spiegel, and S. Pokutta. Projection-Free Adaptive Gradients for Large-Scale Optimization. *preprint*, 10 2020.
- J. Diakonikolas, A. Carderera, and S. Pokutta. Locally Accelerated Conditional Gradients. *Proceedings of AISTATS*, 2020.
- M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.
- R. M. Freund, P. Grigas, and R. Mazumder. An extended Frank-Wolfe method with “in-face” directions, and its application to low-rank matrix completion. *SIAM Journal on Optimization*, 27(1):319–346, 2017.
- G. Gamrath, B. Hiller, and J. Witzig. Reoptimization techniques for mip solvers. In *International Symposium on Experimental Algorithms*, pages 181–192. Springer, 2015.
- D. Garber and E. Hazan. Playing non-linear games with linear oracles. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 420–428. IEEE, 2013.
- D. Garber and O. Meshi. Linear-memory and decomposition-invariant linearly convergent conditional gradient algorithm for structured polytopes. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1001–1009. Curran Associates, Inc., 2016. URL <http://papers.nips.cc/paper/6115-linear-memory-and-decomposition-invariant-linearly-convergent-conditional-gradient-algorithm-for-structured-pdf>.
- E. Hazan and S. Kale. Projection-free online learning. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.
- E. Hazan and H. Luo. Variance-reduced and projection-free stochastic optimization. In *International Conference on Machine Learning*, pages 1263–1271, 2016.
- D. Hendrych, H. Troppens, M. Besançon, and S. Pokutta. Convex integer optimization with frank-wolfe methods. *preprint*, 8 2022.

References II

- M. Jaggi. Revisiting Frank-Wolfe: projection-free sparse convex optimization. In *Proceedings of the 30th International Conference on Machine Learning*, pages 427–435, 2013.
- T. Kerdreux, A. d’Aspremont, and S. Pokutta. Restarting Frank-Wolfe. *Proceedings of AISTATS*, 2019.
- T. Kerdreux, A. d’Aspremont, and S. Pokutta. Projection-Free Optimization on Uniformly Convex Sets. *to appear in Proceedings of AISTATS*, 1 2021a.
- T. Kerdreux, A. d’Aspremont, and S. Pokutta. Local and Global Uniform Convexity Conditions. *preprint*, 2 2021b.
- S. Lacoste-Julien. Convergence rate of Frank-Wolfe for non-convex objectives. *arXiv preprint arXiv:1607.00345*, 2016.
- S. Lacoste-Julien and M. Jaggi. On the global linear convergence of Frank-Wolfe optimization variants. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 496–504. Curran Associates, Inc., 2015. URL <http://papers.nips.cc/paper/5925-on-the-global-linear-convergence-of-frank-wolfe-optimization-variants.pdf>.
- G. Lan and Y. Zhou. Conditional gradient sliding for convex optimization. *SIAM Journal on Optimization*, 26(2):1379–1409, 2016.
- E. S. Levitin and B. T. Polyak. Constrained minimization methods. *USSR Computational Mathematics and Mathematical Physics*, 6(5):1–50, 1966.
- S. J. Reddi, S. Sra, B. Póczos, and A. Smola. Stochastic Frank-Wolfe methods for nonconvex optimization. In *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1244–1251. IEEE, 2016.
- T. Rothvoss. The matching polytope has exponential extension complexity. In *Symposium on Theory of Computing*, pages 263–272, 2014.
- K. Tsuji, K. Tanaka, and S. Pokutta. Pairwise Conditional Gradients without Swap Steps and Sparser Kernel Herding. *To Appear in Proceedings of ICML*, 5 2022.