

# Mixed Integer Convex Programming

## Computational Intelligence, Lecture 11

by Sergei Savin

Fall 2020

# Content

- Mixed Integer Linear Programming (MILP)
- Mixed Integer Quadratic Programming (MIQP)
- Remarks
- Example: Footstep planning
- Big-M method relaxation
  - Basic idea
  - Systems of inequalities
  - Illustration, part 1
  - Illustration, part 2
  - Multiple variables
- Example: Footstep planning
  - Formulation as MIQP
  - Evenly spaced steps
  - Code, part 1
  - Code, part 2
  - Code, part 3
- Homework

# Mixed Integer Linear Programming (MILP)

## General form

A general form of a mixed-integer linear program is:

$$\begin{array}{ll} \underset{\mathbf{x}, \mathbf{y}}{\text{minimize}} & \mathbf{f}_1^\top \mathbf{x} + \mathbf{f}_2^\top \mathbf{y}, \\ \text{subject to} & \left\{ \begin{array}{l} \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \leq \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}, \\ \begin{bmatrix} \mathbf{C}_{11} & \mathbf{C}_{12} \\ \mathbf{C}_{21} & \mathbf{C}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \end{bmatrix}, \\ \mathbf{y} \in \mathbb{N}^n. \end{array} \right. \end{array} \quad (1)$$

In other words, the only difference is that some of the variables (denoted above as  $\mathbf{y}$ ) are only allowed to assume pure integer values.

# Mixed Integer Quadratic Programming (MIQP)

## General form

A general form of a mixed-integer quadratic program is:

$$\begin{aligned} & \underset{\mathbf{x}, \mathbf{y}}{\text{minimize}} && [\mathbf{x} \quad \mathbf{y}] \begin{bmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} \\ \mathbf{H}_{21} & \mathbf{H}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} + [\mathbf{f}_1 \quad \mathbf{f}_2] \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}, \\ & \text{subject to} && \left\{ \begin{array}{l} \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \leq \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}, \\ \begin{bmatrix} \mathbf{C}_{11} & \mathbf{C}_{12} \\ \mathbf{C}_{21} & \mathbf{C}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \end{bmatrix}, \\ \mathbf{y} \in \mathbb{N}^n. \end{array} \right. \end{aligned} \quad (2)$$

Other mixed-integer convex programs can be constructed likewise from their pure convex counterparts.

- Mixed-integer convex programs are not convex, even if the name seems to suggest otherwise.
- The "convex program" in the phrase "mixed-integer convex program" should be understood as "if we fix values of the integer variables, or relax them into reals, the result will be a convex program".
- Mixed integer programs are usually solved using *branch and bound algorithms*; in worse case scenario, the algorithms performs exhaustive search.
- In robotics applications, integer variables in mixed integer programs are often restricted to binary values, i.e.  $\mathbf{y} \in \{0, 1\}^n$ . It is still called "mixed-integer" in the literature, although phrases such as "mixed-binary" or "binary constraints" are not rare.

# Example: Footstep planning

## Problem statement

Given  $N$  convex regions defined by linear inequalities  $\{\mathbf{x} : \mathbf{A}_i \mathbf{x} \leq \mathbf{b}_i\}$  (which is called H-polytope representation), find a sequence of  $K$  points (footsteps) from the given starting point to the given goal point, such that all footsteps lie in one of the convex regions.

$$\begin{aligned} & \underset{\mathbf{x}_1, \dots, \mathbf{x}_K}{\text{minimize}} && \|\mathbf{x}_1 - \mathbf{x}_{\text{start}}\| + \|\mathbf{x}_K - \mathbf{x}_{\text{goal}}\|, \\ & \text{subject to} && \exists \{\mathbf{A}_j, \mathbf{b}_j\} \in \Omega \text{ s.t. } \mathbf{A}_j \mathbf{x}_i \leq \mathbf{b}_j \end{aligned} \tag{3}$$

where  $\Omega = \{\{\mathbf{A}_1, \mathbf{b}_1\}, \{\mathbf{A}_2, \mathbf{b}_2\}, \dots, \{\mathbf{A}_N, \mathbf{b}_N\}\}$ . This is not a convex program.

# Big-M method relaxation

## Basic idea

One of the key methods associated with the use of mixed-integer programming in robotics is the big-M method.

Assume you have two inequalities,  $\mathbf{a}_1^\top \mathbf{x} \leq b_1$  and  $\mathbf{a}_2^\top \mathbf{x} \leq b_2$ , and you are happy if at least one of them holds. Define a new binary variables  $c_1, c_2 \in \{0, 1\}$  and find a big enough constant  $M$ , such that  $\mathbf{a}_1^\top \mathbf{x} \leq b_1 + M$  and  $\mathbf{a}_2^\top \mathbf{x} \leq b_2 + M$  holds for all of your domain (of the part of it you are interested in). Then you write your constraints as:

$$\begin{cases} \mathbf{a}_1^\top \mathbf{x} \leq b_1 + M \cdot c_1 \\ \mathbf{a}_2^\top \mathbf{x} \leq b_2 + M \cdot c_2 \\ c_1 + c_2 = 1 \\ c_i \in \{0, 1\} \end{cases} \quad (4)$$

# Big-M method relaxation

## Systems of inequalities

It works the same way for the case when you have three (or two or more) systems of inequalities  $\mathbf{A}_1\mathbf{x} \leq \mathbf{b}_1$ ,  $\mathbf{A}_2\mathbf{x} \leq \mathbf{b}_2$ ,  $\mathbf{A}_3\mathbf{x} \leq \mathbf{b}_3$  and are happy if at least one holds:

$$\begin{cases} \mathbf{A}_1\mathbf{x} \leq \mathbf{b}_1 + M \cdot \mathbf{1} \cdot (1 - c_1) \\ \mathbf{A}_2\mathbf{x} \leq \mathbf{b}_2 + M \cdot \mathbf{1} \cdot (1 - c_2) \\ \mathbf{A}_3\mathbf{x} \leq \mathbf{b}_3 + M \cdot \mathbf{1} \cdot (1 - c_3) \\ c_1 + c_2 + c_3 = 1 \\ c_i \in \{0, 1\} \end{cases} \quad (5)$$

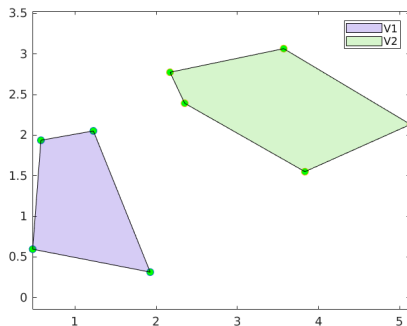
where  $\mathbf{1}$  is a vector of all ones. Notice that constraint  $c_1 + c_2 + c_3 = 1$  can be replaced with  $c_1 + c_2 + c_3 \geq 1$ , allowing avoid relaxing more than one region.



# Big-M method relaxation

## Illustration, part 1

Below are two H-polytopes (convex regions represented by systems of inequalities):

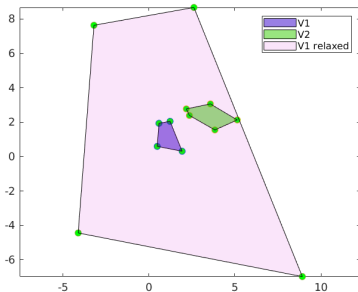
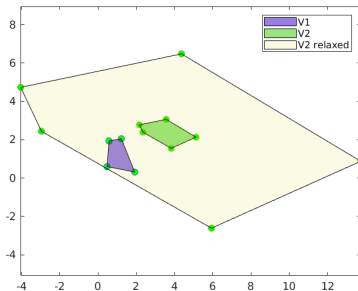


As we can see, their union represents a non-convex domain, and they have no intersection.

# Big-M method relaxation

## Illustration, part 2

Now, one of them is relaxed as described above. Notice that the intersection of the two polytopes is non-relaxed polytope.



# Big-M method relaxation

## Multiple variables

If you have multiple variables  $\mathbf{x}_1, \dots, \mathbf{x}_K$ , and each should belong to at least one of the H-polytopes  $\{\mathbf{A}_i, \mathbf{b}_i\}$ , this can also be represented using big-M method:

$$\begin{cases} \mathbf{A}_1 \mathbf{x}_k \leq \mathbf{b}_1 + M \cdot \mathbf{1} \cdot (1 - c_{1,k}) \\ \mathbf{A}_2 \mathbf{x}_k \leq \mathbf{b}_2 + M \cdot \mathbf{1} \cdot (1 - c_{2,k}) \\ \mathbf{A}_3 \mathbf{x}_k \leq \mathbf{b}_3 + M \cdot \mathbf{1} \cdot (1 - c_{3,k}) \\ c_{1,k} + c_{2,k} + c_{3,k} = 1 \\ c_{i,k} \in \{0, 1\} \\ k = 1, \dots, K \end{cases} \quad (6)$$

Notice that the only difference from the previous example is that now we have  $K$  sets of binary variables  $c_{1,k}$ ,  $c_{2,k}$  and  $c_{3,k}$ .

# Example: Footstep planning

## Formulation as MIQP

Using big-M relaxation we can now formulate the problem as follows:

$$\begin{array}{ll} \underset{\mathbf{x}_1, \dots, \mathbf{x}_K}{\text{minimize}} & \|\mathbf{x}_1 - \mathbf{x}_{\text{start}}\| + \|\mathbf{x}_K - \mathbf{x}_{\text{goal}}\|, \\ \text{subject to} & \begin{cases} \mathbf{A}_i \mathbf{x}_k \leq \mathbf{b}_i + M \cdot \mathbf{1} \cdot (1 - c_{i,k}), \quad i = 1, \dots, N \\ \sum_{i=1}^N c_{i,k} = 1 \\ c \in \{0, 1\}^{N, K} \\ k = 1, \dots, K \end{cases} \end{array} \quad (7)$$

# Example: Footstep planning

## Evenly spaced steps

In order to make the footsteps evenly spaced we add cost on the distance between consequent steps:

$$\begin{aligned} & \underset{\mathbf{x}_1, \dots, \mathbf{x}_K}{\text{minimize}} && \|\mathbf{x}_1 - \mathbf{x}_{\text{start}}\| + \|\mathbf{x}_K - \mathbf{x}_{\text{goal}}\| + w \cdot \sum_{k=1}^{K-1} \|\mathbf{x}_{i+1} - \mathbf{x}_i\|, \\ & \text{subject to} && \begin{cases} \mathbf{A}_i \mathbf{x}_k \leq \mathbf{b}_i + M \cdot \mathbf{1} \cdot (1 - c_{i,k}), \quad i = 1, \dots, N \\ \sum_{i=1}^N c_{i,k} = 1 \\ c \in \{0, 1\}^{N, K} \\ k = 1, \dots, K \end{cases} \end{aligned} \tag{8}$$

where  $w$  is a weight, a coefficient we can tune to adjust relative importance of our primary objective (starting from the point  $\mathbf{x}_{\text{start}}$  and finishing at the point  $\mathbf{x}_{\text{goal}}$  and our secondary objective (making the footsteps evenly spaced).

# Example: Footstep planning

## Code, part 1

```
0 n = 2;
  shift_1 = 1*rand(n, 1);
2 shift_2 = 1*rand(n, 1);
  V1 = randn(n, 6);
4 V2 = randn(n, 6) + shift_1;
  V3 = randn(n, 6) + shift_1 + shift_2;
6
  indices_convhull = convhull(V1');
8 V1 = V1(:, indices_convhull);
  indices_convhull = convhull(V2');
10 V2 = V2(:, indices_convhull);
  indices_convhull = convhull(V3');
12 V3 = V3(:, indices_convhull);

14 [A1, b1] = vert2con(V1');
  [A2, b2] = vert2con(V2');
16 [A3, b3] = vert2con(V3');
```

# Example: Footstep planning

## Code, part 2

```
0 number_of_steps = 7;
  start_point = sum(V1, 2) / size(V1, 2);
2 finish_point = sum(V3, 2) / size(V3, 2);
  weight_goal = 5;
4 bigM = 15;

6 cvx_begin
    variable x(n, number_of_steps)
    binary variable c(3, number_of_steps);

10    cost = 0;
    for i = 1:(number_of_steps-1)
12        cost = cost + norm(x(:, i) - x(:, i+1));
    end
14    cost = cost + norm(x(:, 1) - start_point)*
    weight_goal;
    cost = cost + norm(x(:, number_of_steps) -
    finish_point)*weight_goal;
16    minimize( cost )
```

# Example: Footstep planning

## Code, part 3

```
0      subject to
1          for i = 1:number_of_steps
2              A1*x(:, i) <= b1 + (1 - c(1, i))*bigM;
3              A2*x(:, i) <= b2 + (1 - c(2, i))*bigM;
4              A3*x(:, i) <= b3 + (1 - c(3, i))*bigM;
5              c(1, i) + c(2, i) + c(3, i) == 1;
6          end
7      cvx_end
8
9      plot(x(1, :)', x(2, :)', '^', 'MarkerEdgeColor', 'k', '
10           MarkerSize', 10, 'LineWidth', 2); hold on;
```



Implement footstep planning for a biped, making sure every pair of steps lands in the same H-polytope.

Lecture slides are available via Moodle.

You can help improve these slides at:

[github.com/SergeiSa/Computational-Intelligence-Slides-Fall-2020](https://github.com/SergeiSa/Computational-Intelligence-Slides-Fall-2020)

Check Moodle for additional links, videos, textbook suggestions.