Mixed Integer Convex Programming

Computational Intelligence, Lecture 12

by Sergei Savin

Spring 2024

Content

- Mixed Integer Linear Programming (MILP)
- Mixed Integer Quadratic Programming (MIQP)
- Example: Footstep planning
- Big-M method relaxation
- Example: switching control
- Homework

Mixed Integer Linear Programming (MILP)

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

$$\begin{aligned} & \underset{\mathbf{x}}{\text{minimize}} && \mathbf{f}^\top \mathbf{x}, \\ & \text{subject to} && \begin{cases} \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\ \mathbf{C}\mathbf{x} = \mathbf{d}, \\ \mathbf{x}_1, ..., \mathbf{x}_m \in \mathbb{R}, \\ \mathbf{x}_{m+1}, ..., \mathbf{x}_n \in \mathbb{N}. \end{cases} \end{aligned} \tag{1}$$

In other words, the only difference is that some of the variables are only allowed to assume pure integer values.

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

$$
\begin{aligned}
&\underset{\mathbf{x}}{\text{minimize}} \quad l(\mathbf{x}) \quad \text{convex cost} \\
&\text{subject to} \quad
\begin{cases}
g(\mathbf{x}) \leq 0, & \text{convex domain} \\
\mathbf{C}\mathbf{x} = \mathbf{d}, & \\
x_1, ..., x_m \in \mathbb{R}, & \text{continuous variables} \\
x_{m+1}, ..., x_n \in \mathbb{N}. & \text{integer variables}
\end{cases}
\end{aligned}
\tag{2}
$$

Remarks

- 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.

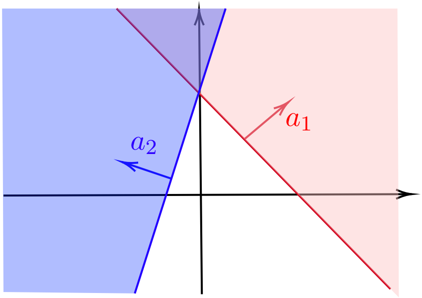
Given N convex regions defined by linear inequalities $\{\mathbf{x} : \ \mathbf{A}_i \mathbf{x} \le \mathbf{b}_i\}$ (which is called H-polytope representation), find a sequence of K points (footsteps) from the given starting point to the given goal point, such that all footsteps lie in one of the convex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

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

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.

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).



Big-M method, 2

Let us put the following constraint on the variables c_1, c_2:

$$c_1 + c_2 = 1 \tag{4}$$

The pair of variables (c_1, c_2) together can assume only two values - (1, 0) and (0, 1). We can modify the inequalities $\mathbf{a}_1^\top \mathbf{x} \leq b_1$ and $\mathbf{a}_2^\top \mathbf{x} \leq b_2$:

$$\begin{cases} \mathbf{a}_1^\top \mathbf{x} \leq b_1 + M \cdot c_1 \\ \mathbf{a}_2^\top \mathbf{x} \leq b_2 + M \cdot c_2 \end{cases} \tag{5}$$

If $(c_1, c_2) = (1, 0)$, then modified inequalities become:

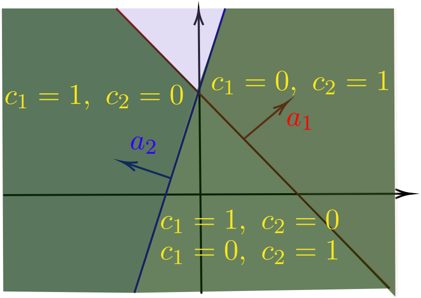
$$\begin{cases} \mathbf{a}_1^\top \mathbf{x} \leq b_1 + M & \text{- true for all } \mathbf{x} \\ \mathbf{a}_2^\top \mathbf{x} \leq b_2 & \text{- original constraint} \end{cases} \tag{6}$$

If $(c_1, c_2) = (0, 1)$, then modified inequalities become:

$$\begin{cases} \mathbf{a}_1^\top \mathbf{x} \leq b_1 & \text{- original constraint} \\ \mathbf{a}_2^\top \mathbf{x} \leq b_2 + M & \text{- true for all } \mathbf{x} \end{cases} \tag{7}$$

With that, we propose the following constraint:

$$\begin{cases} \mathbf{a}_1^\top \mathbf{x} \leq b_1 + M \cdot c_1 \\ \mathbf{a}_2^\top \mathbf{x} \leq b_2 + M \cdot c_2 \\ c_1 + c_2 = 1 \\ c_i \in \{0,\ 1\} \end{cases} \tag{8}$$



Choosing between polytopes, 1

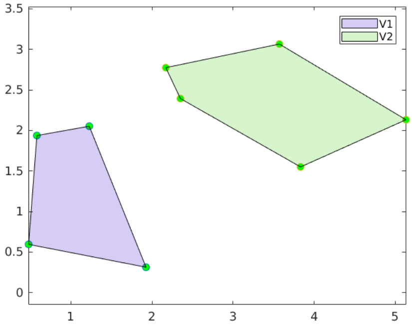
It works the same way for the case when you have three (or two or more) systems of inequalities $\mathbf{A}_1\mathbf{x} \leq \mathbf{b}_1$, $\mathbf{A}_2\mathbf{x} \leq \mathbf{b}_2$, $\mathbf{A}_3\mathbf{x} \leq \mathbf{b}_3$ and are happy if at least one holds:

$$\begin{cases} \mathbf{A}_1\mathbf{x} \leq \mathbf{b}_1 + M \cdot \mathbf{1} \cdot (1 - c_1) \\ \mathbf{A}_2\mathbf{x} \leq \mathbf{b}_2 + M \cdot \mathbf{1} \cdot (1 - c_2) \\ \mathbf{A}_3\mathbf{x} \leq \mathbf{b}_3 + M \cdot \mathbf{1} \cdot (1 - c_3) \\ c_1 + c_2 + c_3 = 1 \\ c_i \in \{0,\ 1\} \end{cases} \tag{9}$$

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

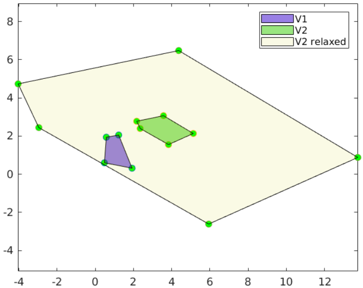
Choosing between polytopes, 2

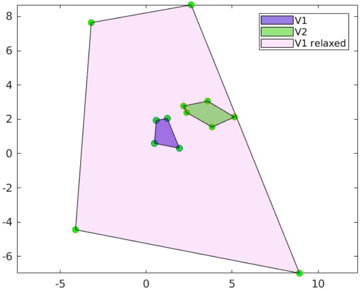
Below are two H-polytops (convex regions represented by systems of inequalities):



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

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





Choosing between polytopes, 4

Multiple variables

If you have multiple variables $\mathbf{x}_1$, ..., $\mathbf{x}_K$, 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:

$$\begin{cases} \mathbf{A}_1 \mathbf{x}_k \leq \mathbf{b}_1 + M \cdot \mathbf{1} \cdot (1 - c_{1,k}) \\ \mathbf{A}_2 \mathbf{x}_k \leq \mathbf{b}_2 + M \cdot \mathbf{1} \cdot (1 - c_{2,k}) \\ \mathbf{A}_3 \mathbf{x}_k \leq \mathbf{b}_3 + M \cdot \mathbf{1} \cdot (1 - c_{3,k}) \\ c_{1,k} + c_{2,k} + c_{3,k} = 1 \\ c_{i,k} \in \{0,\ 1\} \\ k = 1, ..., K \end{cases} \tag{10}$$

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 $c_{3,k}$.

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

$$
\begin{aligned}
&\underset{\mathbf{x}_k,\ \mathbf{c}_{i,k}}{\text{minimize}} && ||\mathbf{x}_1 - \mathbf{x}_{\text{start}}|| + ||\mathbf{x}_K - \mathbf{x}_{\text{goal}}||, \\
&\text{subject to} && \begin{cases} \mathbf{A}_i \mathbf{x}_k \leq \mathbf{b}_i + M \cdot \mathbf{1} \cdot (1 - c_{i,k}), \ \ i = 1, ..., N \\ \sum_{i=1}^{N} c_{i,k} = 1 \\ c \in \{0,\ 1\}^{N,\ K} \\ k = 1, ..., K \end{cases}
\end{aligned}
\tag{11}
$$

Example: Footstep planning

Evenly spaced steps

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

$$\underset{\mathbf{x}_k,\ \mathbf{c}_{i,k}}{\text{minimize}} \quad ||\mathbf{x}_1 - \mathbf{x}_{\text{start}}||^2 + ||\mathbf{x}_K - \mathbf{x}_{\text{goal}}||^2 + w \cdot \sum_{k=1}^{K-1} ||\mathbf{x}_{i+1} - \mathbf{x}_i||^2,$$

$$\text{subject to} \quad \begin{cases} \mathbf{A}_i \mathbf{x}_k \le \mathbf{b}_i + M \cdot \mathbf{1} \cdot (1 - c_{i,k}),\ \ i = 1, ..., N \\ \sum_{i=1}^{N} c_{i,k} = 1 \\ c \in \{0,\ 1\}^{N,\ K} \\ k = 1, ..., K \end{cases}$$

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

```
n = 2;
shift_1 = 1*rand(n, 1);
shift_2 = 1*rand(n, 1);
V1 = randn(n, 6);
V2 = randn(n, 6) + shift_1;
V3 = randn(n, 6) + shift_1 + shift_2;

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

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

Example: Footstep planning

Code, part 2

```
cvx_begin
    variable x(n, number_of_steps)
    binary variable c(3, number_of_steps);
    cost = 0;
    for i = 1:(number_of_steps-1)
        cost = cost + norm(x(:, i) - x(:, i+1));     end
    cost = cost + (norm(x(:, 1) - start_point) + norm(x(:, number_of_steps) - finish_point))*weight;
    minimize( cost )
    subject to
for i = 1:number_of_steps
  A1*x(:, i) <= b1 + (1 - c(1, i))*bigM;
  A2*x(:, i) <= b2 + (1 - c(2, i))*bigM;
  A3*x(:, i) <= b3 + (1 - c(3, i))*bigM;
  c(1, i) + c(2, i) + c(3, i) == 1;
end
cvx_end
plot(x(1, :)', x(2, :)', '^', 'MarkerEdgeColor', 'k', 'MarkerSize', 10, 'LineWidth', 2); hold on;
```

Consider the case when you have a control system $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}_1\mathbf{u}_1 + \mathbf{B}_2\mathbf{u}_2$, but you are required to use either $\mathbf{B}_1\mathbf{u}_1$ or $\mathbf{B}_2\mathbf{u}_2$ at any time, not both.

This can be cast as a mixed-integer constraint:

$$\begin{cases} ||\mathbf{u}_1|| \leq M(1 - c_1) \\ ||\mathbf{u}_2|| \leq M(1 - c_2) \\ c_1 + c_2 = 1 \end{cases} \tag{12}$$

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

Lecture slides are available via Github, links are on Moodle:

github.com/SergeiSa/Computational-Intelligence-2024