Lecture 13 2025-10-07

Today: mixed-integer programs

So far, the taxonomy of optimization problems has been:

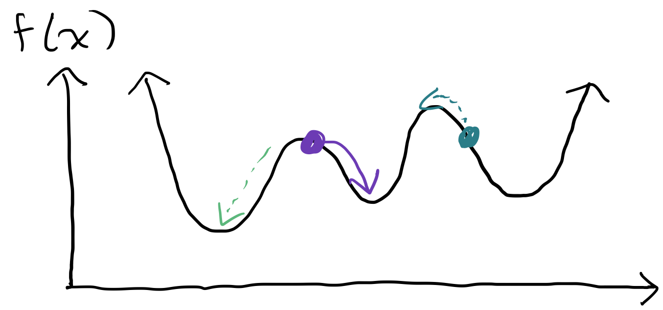
1. Unconstrained vs. constrained
2. Convex vs. non-convex
3. Smooth vs. non-smooth
4. Continuous vs. discrete } Today
5. Deterministic vs. stochastic

Local vs. global methods:

Local methods: so far, we've focused on local methods; given an initial point $x^{(0)}$, iteratively take "small" steps $\Delta x^{(k)}$ until necessary conditions of optimality are satisfied.

Pro: simple to implement and debug, computationally
inexpensive (have polynomial time algorithms for many convex algorithms)

Con: sensitive to initial guess $x^{(0)}$ and can get stuck in local minima for non-convex problems

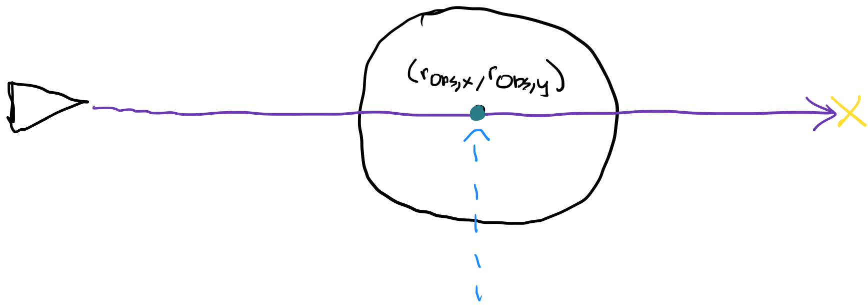


e. g. Obstacle avoidance problems: in

"pathological" cases, cannot get unstuck

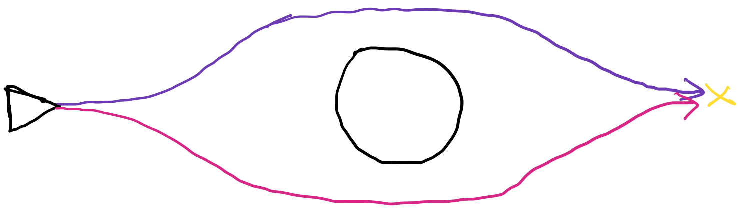
constraint: $(x - r_{obs,x})^2 + (y - r_{obs,y})^2 \geq r_{min}^2$

$\rightarrow$ gradient: $\begin{pmatrix} -(x - r_{obs,x}) \\ -(y - r_{obs,y}) \end{pmatrix}$



gradient $\nabla g = 0$ here!

combinatorial choices: have two equally valid global optima to choose from:



Global methods: searches over full space to find globally optmal solution

We'll cover three global search techniques in the coming weeks:

1. Integer programs

2. Approximate methods

3. Random search

Integer programming:

$$\min_{x_{0:N},\, u_{0:N},\, z_{0:N}} \sum_{k=1}^{N} g_k(x_k, u_k, z_k)$$

$$\text{subj. to: } x_{k+1} = f(x_k, u_k, z_k) \qquad k = 0, \dots, N-1$$

$$g(x_k, u_k, z_k) \leq 0 \qquad k = 0, \ldots, N$$

$x_k \in \mathbb{R}^{n_x}$

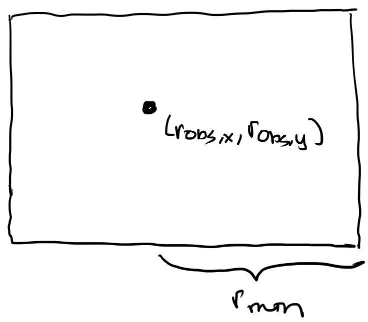
$u_k \in \mathbb{R}^{n_u}$

$z_k \in \mathbb{Z}^{n_z}$ ← used to capture combinatorial or logical constraints in decision making

Without loss of generality, we'll work with binary decision variables $\delta_k \in \{0, 1\}^{n_z}$ as we can reformulate integer and binary programs with one another

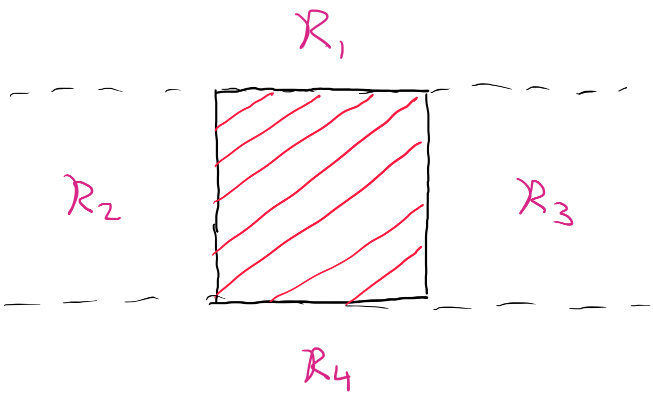
Mixed integer programs (MIPs) are used to model

Collision avoidance: $x \notin X_{obs}$



constraint: $\|(x - r_{obs,x},\ y - r_{obs,y})\|_{\ell_1} \geq r_{min}$

using integer programming, we can rewrite this as a disjunctive constraint



$$x \notin X_{obs} \iff x \in R_1 \vee x \in R_2 \vee x \in R_3 \vee x \in R_4$$

Let's consider each region's constraints:

R_1: $x_{R,y} \geq r_{obs,y} + r_{min} \rightarrow (r_{obs,y} + r_{min}) - x_{R,y} \leq 0 \quad M(1-\delta^{(1)})$

R_2: $x_{R,x} - (r_{obs,x} - r_{min}) \leq 0 \quad M(1-\delta^{(2)})$

$x_{R,y} - (r_{obs,y} + r_{min}) \leq 0 \quad M(1-\delta^{(2)})$

$$(r_{obs,y} - r_{min}) - x_{R,y} \leq 0 \quad M(1-\delta^{(2)})$$

$$\boxed{\sum_{i=1}^{4} \delta^{(i)} = 1}$$

R_3: $$(r_{obs,x} + r_{min}) - x_{R,x} \leq 0 \quad M(1-\delta^{(3)})$$

$$x_{R,y} - (r_{obs,y} + r_{min}) \leq 0 \quad M(1-\delta^{(3)})$$

$$(r_{obs,y} - r_{min}) - x_{R,y} \leq 0 \quad M(1-\delta^{(3)})$$

R_4: $$x_{R,y} - (r_{obs,y} - r_{min}) \leq 0 \quad M(1-\delta^{(4)})$$

We can use the big-M approach to enforce the disjoint constraint where $M >> 0$ is some sufficiently large number

→ introduce a binary variable $\delta^{(i)}$ for each region above, where $\delta^{(i)} \in \{0, 1\}$

The key constraint is then: $\sum_{i=1}^{4} \delta^{(i)} = 1$

Note that if $\delta^{(i),*} = 0$, then the inequality it's used on is trivially satisfied:

e.g., For R_4, if $\delta^{(4),*} = 0$, then:

$$x_{R,y}^* - (r_{obs,y} - r_{min}) \leq M$$

so any value of $x_{R,y}^*$ trivially satisfies the constraint with $\delta^{(4),*} = 0$ plugged in

Alternatively, a more succinct set of constraints is:

$$(r_{obs,x} + r_{min}) - M\,\delta^{(1)} \le x_{R,x} \le (r_{obs,x} - r_{min}) - M\,\delta^{(2)}$$

$$(r_{obs,y} + r_{min}) - M\,\delta^{(3)} \le x_{R,y} \le (r_{obs,y} - r_{min}) - M\,\delta^{(4)}$$

$$\sum_{i=1}^{4} \delta^{(i)} \le 3$$

$$\delta^{(i)} \in \{0, 1\}$$

Piecewise affine dynamics

Suppose we have N_m modes of dynamics to switch between,

$$\{A^i, B^i\}_{i=1}^{N_m}$$

using big-M notation, we can enforce this as:

$$x_{k+1} - (A^i x_k + B^i u_k) \leq M(1 - \delta^{(i)})$$
$$(A^i x_k + B^i u_k) - x_{k+1} \leq M(1 - \delta^{(i)})$$

$$\sum_{i=1}^{N_m} \delta^{(i)} = 1$$

Mixed-integer convex programs (MICPs): if $\mathcal{P}$ is convex w.r.t. x and u

Mixed-integer nonlinear programs (MINLPs): $\mathcal{P}$

not necessarily convex w.r.t. x and u

MINLPs / MICPs:

Pros:

- powerful & expressive modeling formalism
- captures many task planning and logical constraints

Con:

- worst-case exponential complexity $O(2^{n_z})$
- far fewer solver options available

For MICPs, for a "reasonable" n_z, there exist algorithms that find globally optimal solutions far faster

↳ branch-and-bound, branch-and-cut, Bender's decomposition

Branch-and-bound

Tree-search based approach where each node solves a convex relaxation of the MICP and uses "pruning" rules to avoid searching all combinatorial assignments

underlying idea: if optimization problem $\mathcal{P}$ is an MICP where $z_i \in \{0,1\}$, then relaxing this constraint to $z_i \in [0,1]$ yields a convex relaxation $\overline{\mathcal{P}}$

→ $\mathcal{P}$: MICP with $z_i \in \{0,1\}$

$\bar{P}$: CP with $z_i \in [0,1]$

What can we say about the optimal value J^* for P and the optimal value $\bar{J}^*$ for $\bar{P}$?

$$\Rightarrow \bar{J}^* \leq J^*$$

since $\bar{P}$ has a "larger" feasible set

Key idea behind B&B: track upper and lower bounds for J^* and prune nodes for sub-trees that cannot yield an improvement to the solution

Track J^{LB} and J^{UB}, such that

$$J^{LB} \leq J^* \leq J^{UB}$$

terminate when $|J^{LB} - J^{UB}| \leq \varepsilon$

How to get J^{LB} & J^{UB}?

J^{LB}: at each node in B&B, $J^{LB} = \bar{J}^*$, where $\bar{J}^*$ is the cost of the convex relaxation at that problem

J^{UB}: if an integer feasible solution (i.e., $z \in \{0,1\}^{n_z}$) exists, then this upper bounds J^* as it is a feasible but not necessarily optimal solution

Steps:

1. Solve relaxed $\bar{P}$ with $z \in [0,1]^{n_z}$ and set $J^{LB} = \bar{J}^*$

   Set $J^{UB} = \infty$ if no other integer feasible solution exists

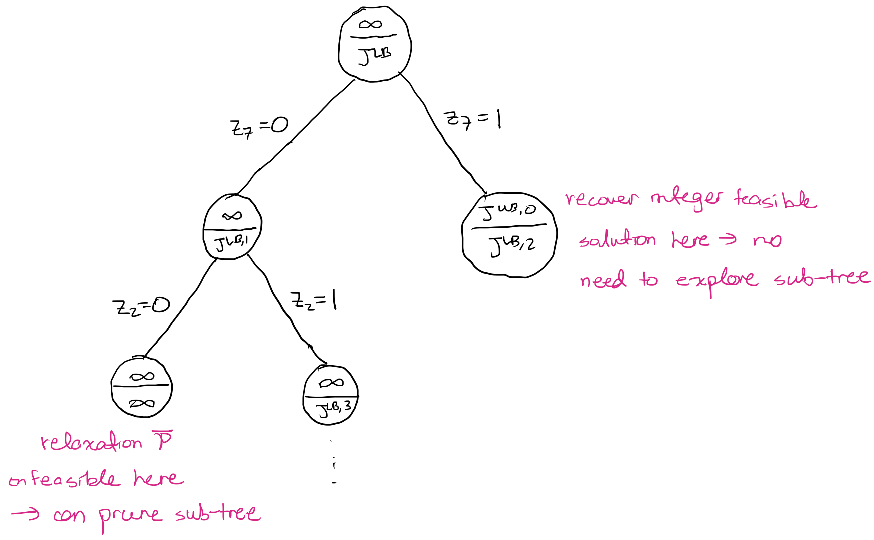
Set $J^{UB} = \infty$ if no other integer

2. Branch on a variable z_i and create two sub-trees with $z_i = 0$ and $z_i = 1$ each.

3. Solve relaxed problem at each node

4. Update J^{LB} / J^{UB} at each node

5. Iterate



Three pruning rules:

1. If relaxation $\bar{P}$ is infeasible
   ↳ searching sub-tree entails solving more constrained problems, so cannot possibly yield feasible problem

2. If relaxation $\bar{P}$ yields integral solution
   ↳ if relaxed soln. has integer value, then no need to branch further

3. If relaxation $\bar{P}$ has a cost $J^* \geq J^{UB}$
   ↳ relaxed problem attains worse cost than a feasible solution we already have