16.410/413
Principles of Autonomy and Decision Making

Lecture 18: (Mixed-Integer) Linear Programming for Vehicle Routing and Motion Planning

Emilio Frazzoli

Aeronautics and Astronautics
Massachusetts Institute of Technology

November 15, 2010

Assignments

Readings

- Lecture notes
- [IOR] Chapter 11.

(Mixed) Integer Linear Programming

- Many problems of interest can be formulated as mathematical programs in which some of the decision variables are constrained to take one of a finite set of values
- Typically, these represent logical decisions: visit a location or not, do a task before another, pass to the left or to the right of an obstacle, etc.
- These can often be modeled as "LP's", in which some of the variables must take discrete values (or binary 0/1 values.)
- Let us look at some examples.

Vehicle Routing Problems

- We have already studied a basic problem in robotics and automation, i.e., the computation of a shortest path between a start and a goal location.
- In many applications, e.g., UAV mission planning problems, it is of interest to compute paths for one or more vehicles to reach a number of locations, while optimizing some performance criterion.
- Vehicle Routing Problems are essentially shortest path problems for multiple vehicles and/or multiple destinations, subject to a variety of constraints or performance objectives.
- VRPs come in a large number of varieties, we will look at some examples.

The Traveling Salesman Problem

- The Traveling Salesman Problem (TSP) is an example of a VRP, in which a single vehicle must visit N locations (cities) following a minimum-cost closed path, starting and ending at a depot.
- Formal definition, as a graph problem:
  Let $G = (V, E, w)$ be a complete undirected weighted graph, whose vertices include the depot V_0, and the locations to be visited $V_1, \ldots, V_N$. Compute a minimum-weight Hamiltonian cycle for G (i.e., a closed path through all vertices).
- Prototypical hard combinatorial problem (NP-hard).
  (But polynomial-time approximations exist for metric TSPs, i.e., TSPs in which the weights satisfy the triangle inequality!)
- It is possible to write the TSP in a LP form...

A naïve LP formulation

- Binary decision variables x_e, $e \in E$: $x_e = 1$ if the path includes edge e, and $x_e = 0$ otherwise.
- Let S be a proper subset of V, i.e., $\emptyset \subset S \subset V$, and indicate with $\delta(S) \subset E$ the set of edges that have exactly one endpoint in V.
- It is tempting to formulate the problem as

$$\begin{aligned} \min \quad & \sum_{e \in E} w(e)\, x_e \\ \text{s.t.:} \quad & \sum_{e \in \delta(\{v\})} x_e = 2, \qquad \forall v \in V \\ & 0 \leq x_e \leq 1, \qquad \forall e \in E. \end{aligned}$$

- What can go wrong?

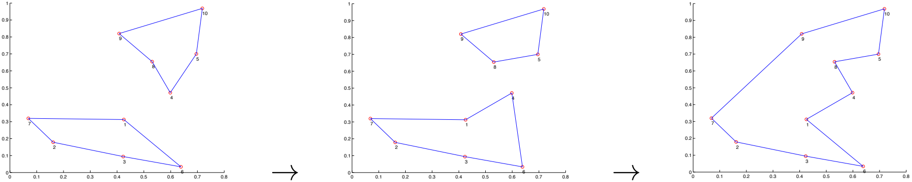
Sub-tour elimination

- In general, it may happen that
  - the values x_e are not 0 or 1 (unlike in the shortest path problem, the constraint matrix in the LP is not totally unimodular).
  - the edges such that $x_e > 0$ may form more than one (sub)tour.
- So it is necessary to add integrality constraints, and sub-tour elimination constraints:

$$
\begin{aligned}
\text{min} \quad & \sum_{e \in E} w(e)\, x_e \\
\text{s.t.:} \quad & \sum_{e \in \delta(\{v\})} x_e = 2, \qquad \forall v \in V \\
& \sum_{e \in \delta(S)} x_e \geq 2, \qquad \forall S \subset V, \mathrm{card}(S) \geq 3 \\
& x_e \in \{0, 1\}, \qquad \forall e \in E.
\end{aligned}
$$

Sub-tour elimination in practice

- Sub-tour elimination constraints are exponentially many.
- In practice, one can attempt to solve the problem without sub-tour elimination constraints. If the solution contains subtours, add constraints eliminating those subtours, and repeat.
- In each case, if the integrality constraint is relaxed, the problem is a LP. If the solution of the LP is integral and contains no subtour, that solution is optimal.

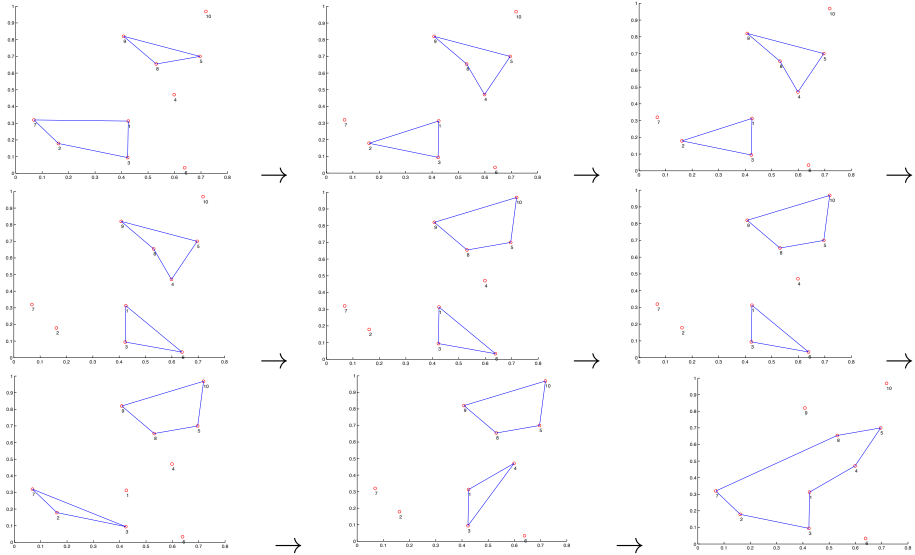


Skipping cities in TSP

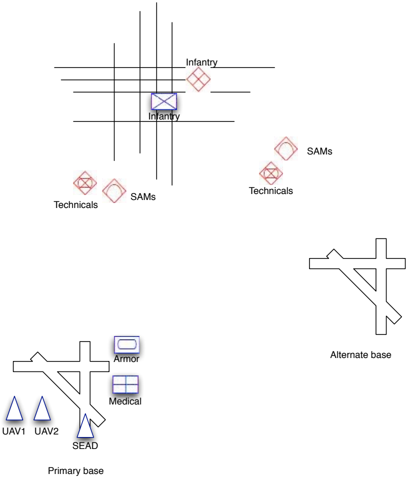
- What if there are options allowing the moving agent to visit only a subset of all the points? E.g., it is ok to skip n_s of the given points, for some $n_\mathrm{s} < N$.
- Introduce a new binary variable b_v for each vertex: $b_v = 1$ if the tour "skips" vertex v, and $b_v = 0$ otherwise.
- Then, one can write the problem as

$$
\begin{aligned}
\min \quad & \sum_{e \in E} w(e)\, x_e \\
\text{s.t.:} \quad & \sum_{e \in \delta(\{v\})} x_e = 2 - 2b_v, \qquad \forall v \in V, \\
& \sum_{e \in \delta(S)} x_e \geq 2 - 2 \sum_{v \in S} b_v, \qquad \forall S \subset V, \mathrm{card}(S) \geq 3, \\
& \sum_{v \in V} b_v = n_\mathrm{s}, \\
& x_e \in \{0, 1\}, \qquad \forall e \in E, \\
& b_v \in \{0, 1\}, \qquad \forall v \in V.
\end{aligned}
$$

Example



A Complex multi-UAV Mission



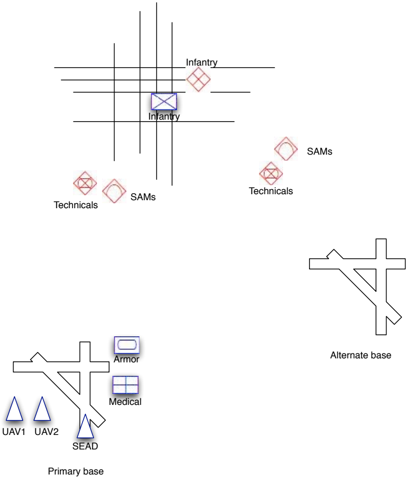
Mission specs

- Infantry unit pinned down by insurgents in an urban area.
- Egress routes blocked by technicals, protected by SAM units.
- Help infantry unit to reach a base with a medic in minimum time/minimum total flight time.

Friendly units

- Two UAVs capable of taking out ground targets, but vulnerable to SAMs.
- One SEAD UAV.
- One armored unit.
- One medical unit.

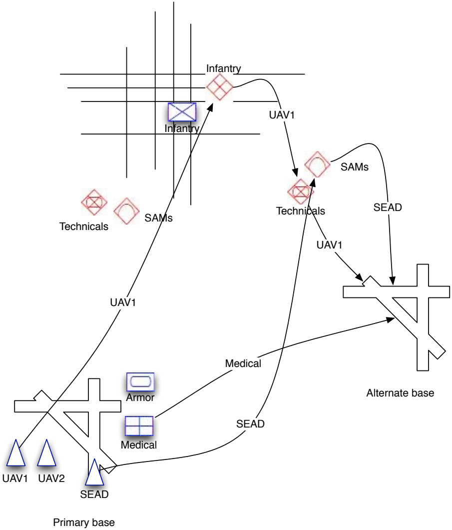
A Complex multi-UAV Mission



In LTL

- Attack enemy infantry $\diamond P_{\text{infantry}}$
- Attack either Technical 1 or 2 $\diamond(P_{\text{Tech1}} \wedge P_{\text{Tech2}})$.
- UAV1 and UAV2 cannot engage Technical 1, unless SAM site 1 has been destroyed: $\neg(S_{\text{Tech1,UAV1}-2})\mathcal{W}(P_{\text{SAM1}})$.
- UAV1 and UAV2 cannot engage the SAM sites at al: $\square(\neg S_{\text{SAM1,UAV1}} \wedge \neg S_{\text{SAM1,UAV2}} \wedge \ldots)$
- ...

Optimal solution



A class of Trajectory Optimization problems

Vehicle Dynamics

Consider a dynamical system described by equations of the form:

$$\frac{d}{dt}x(t) = f(x(t), u(t)), \quad x(0) = x_0.$$

Trajectory optimization

Common trajectory optimization problems take the form
"compute the optimal input function u such that:

- *a target point is reached within a given time (or in minimum time);*
- *the total control effort (e.g., fuel burned) is minimized;*
- *the instantaneous control effort is bounded by $u_{\max}$;*
- *the maximum speed is bounded by $v_{\max}$;*
- *the vehicle remains within some given (convex) boundaries."*

Mathematical Formalization

All these problems can be stated as **optimal control** problems, of the form:

$$
\begin{aligned}
\min_{u(\cdot)} \quad & \Gamma(x(T)) + \int_0^T \gamma(x(t), u(t))\, dt, \\
\text{s.t.:} \quad & \frac{d}{dt}x(t) = f(x(t), u(t)), \quad \forall t \in [0, T], \\
& g(x(t), u(t)) \leq 0., \quad \forall t \in [0, T].
\end{aligned}
$$

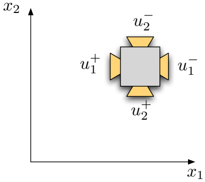
Linear Programming for trajectory optimization

- If a numerical solution is sought, usually the optimal control problem is discretized, e.g., assuming that u is a piecewise-constant function. *Instead of looking for an optimal* function *(infinite-dimensional object), write the problem in terms of a finite number of decision variables.*
- If all the functions appearing in the optimal control problems are linear (or can be approximated by a linear function), then the discretized problem can be written as an LP:

$$\begin{aligned}
\min_{(u[0],u[1],\ldots u[N])} \quad & C^T x[N] + \sum_{i=0}^{N} \left(c^T x[i] + d^T u[i] \right), \\
\text{s.t.:} \quad & x[i+1] = Ax[i] + Bu[i], \quad \forall i \in \{0, \ldots, N\}, \\
& g^T x[i] + h^T u[i] \leq m, \quad \forall i \in \{0, \ldots, N\}.
\end{aligned}$$

Example: planar spacecraft with 4 thrusters

- Consider a square spacecraft moving on a plane, in **deep space**.
- The spacecraft is equipped with 4 thrusters, each firing on one side of the spacecraft, along a line aligned with the spacecraft's center of mass.
- The spacecrafts dynamics are well modeled by a double integrator:



$$\frac{d^2}{dt^2}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} u_1^+(t) - u_1^-(t) \\ u_2^+(t) - u_2^-(t) \end{bmatrix}.$$

- Integration of the above differential equations, assuming a zero-order hold on the control inputs, yields:

$$\underbrace{\begin{bmatrix} x_1(t+\Delta t) \\ x_2(t+\Delta t) \\ \dot{x}_1(t+\Delta t) \\ \dot{x}_2(t+\Delta t) \end{bmatrix}}_{x[i+1]} = \underbrace{\begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{A_\mathrm{d}} \underbrace{\begin{bmatrix} x_1(t) \\ x_2(t) \\ \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix}}_{x[i]} + \underbrace{\begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 & -\frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 & 0 & -\frac{1}{2}\Delta t^2 \\ \Delta t & 0 & -\Delta t & 0 \\ 0 & \Delta t & 0 & -\Delta t \end{bmatrix}}_{B_\mathrm{d}} \underbrace{\begin{bmatrix} u_1^+(t) \\ u_2^+(t) \\ u_1^-(t) \\ u_2^-(t) \end{bmatrix}}_{u[i]}.$$

Example: LP formulation

- It is desired to reposition the spacecraft to the origin at rest in N steps, using minimum fuel.
- Objective: $\min_u \sum_{i=0}^{N}[1,1,1,1]u[i] = \underbrace{[1,1,\ldots,1]}_{4\times(N+1)} \begin{bmatrix} u[0] \\ \hline u[1] \\ \hline \ldots \\ \hline u[N] \end{bmatrix}.$
- Terminal constraint:

$$\begin{aligned} x[N] &= A_\mathrm{d}x[N-1] + B_\mathrm{d}u[N-1] \\ &= A_\mathrm{d}^2x[N-2] + A_\mathrm{d}B_\mathrm{d}u[N-2] + B_\mathrm{d}u[N-1] = \ldots \\ &= A_\mathrm{d}^N x[0] + [A_\mathrm{d}^{N-1}B_\mathrm{d}, A_\mathrm{d}^{N-2}B_\mathrm{d}, \ldots, B_\mathrm{d}] \begin{bmatrix} u[0] \\ \hline u[1] \\ \hline \ldots \\ \hline u[N] \end{bmatrix} = 0. \end{aligned}$$

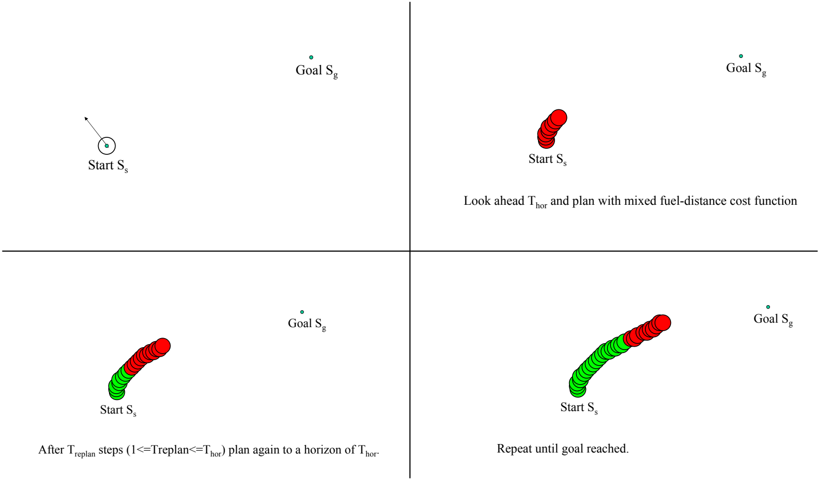
- Thrust magnitude bounds: $u[i] \le u_\mathrm{max}, \quad \forall i \in \{0,1,\ldots,N\}$.
- Non-negativity constraints: $u[i] \ge 0, \quad \forall i \in \{0,1,\ldots,N\}$.

Example: Receding Horizon Strategy

What if N steps are not sufficient to reach the target?

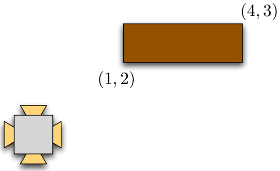
- Add a terminal cost, weighing the distance from the origin.
  - New cost: $d + \mathbf{1}^T u$.
  - Relax terminal constraints: $$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & k & 0 \\ 0 & 0 & 0 & k \\ 0 & 0 & -k & 0 \\ 0 & 0 & 0 & -k \end{bmatrix} x[N] \leq d\mathbf{1}$$
- Receding horizon implementation:
  - Plan for N steps;
  - Execute the first $n < N$;
  - Iterate (i.e., plan for for another N steps, etc.)

Receding horizon output



Avoiding static obstacles

- What if there is an obstacle (i.e., the space station) in the path of the spacecraft?
  - Need to enforce collision avoidance constraints.
- Collision avoidance constraints are not convex, and they cannot be written as a LP.
  - In a LP, all constraints are and, i.e., they all must hold at the same time.
- Similar conclusions hold for moving obstacles, plume impingement constraints, etc., as well as for multi-vehicle collision avoidance.



$$
\begin{aligned}
x_1[i] &\le 1, \text{ or} \\
x_1[i] &\ge 4, \text{ or} \\
x_2[i] &\le 2, \text{ or} \\
x_2[i] &\ge 3.
\end{aligned}
$$

$$(\forall i \in \{0, \ldots, N\})$$

Reformulation as MILP

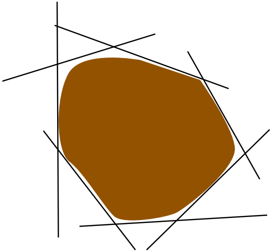
- It is possible to rewrite the "or" collision avoidance constraints in the form of "and" constraints, e.g., (M is a "large number").

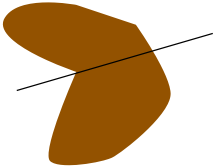
$$
\begin{aligned}
x_1[i] &\leq 1 + Mb_1[i], \text{ and} \\
-x_1[i] &\leq -4 + Mb_2[i], \text{ and} \\
x_2[i] &\leq 2 + Mb_3[i], \text{ and} \\
-x_2[i] &\leq -3 + Mb_4[i], \text{ and} \\
b_1[i] + b_2[i] + b_3[i] + b_4[i] &\leq 3, \text{ and} \\
b_1[i], b_2[i], b_3[i], b_4[i] &\in \{0, 1\} \\
& (\forall i \in \{0, \ldots, N\})
\end{aligned}
$$

- The "or" trouble has been moved to the binary variables b, which can only take the value 0 or the value 1.
- Apart from the binary variables, the rest of the problem "looks like a LP."
- This leads to a case of **Mixed-Integer Linear Program (MILP)**.

Arbitrarily shaped obstacles

- If you have obstacles of arbitrary shapes, you can approximate their convex hull arbitrarily well with linear constraints. At least one of them must be applied ("or")
- Non-convex obstacles can be split up into convex pieces, and the same technique can be applied.
- In 3d, the lines become planes, polygons become polyhedra, but the idea remains the same.





Mixed-Integer Linear Programs

- The general form of a MILP is the following:

$$\begin{aligned} \min_x \quad & c^T x + d^T z \\ \text{s.t.:} \quad & Ax + Bz \leq b \\ & x \geq 0 \\ & z \in \{0, 1\}^{N_z} \end{aligned}$$

- Looks like a regular LP, with the difference that at least some of the decision variables are constrained to integer values (or, without loss of generality, Boolean/binary values).
- MILPs can approximate a very large class of problems (including nonlinear, non-convex, optimization problems), in particular including problems with logical variables.
  (E.g., pass to the left OR to the right of an obstacle, visit target A or target B, etc.)

Complexity of BIPs

- Consider the following Binary Integer Program:

$$\begin{aligned} \min_x \quad & d^T z \\ \text{s.t.:} \quad & Bz \leq b \\ & z \in \{0,1\}^{N_z} \end{aligned}$$

- Is it more or less difficult to solve than a similar LP?
  - In principle, there are only a finite number of possible solutions...
  - However, there are 2^{N_z} of them!
- In general, IPs (and BIPs, and MILPs) require exponential time to solve.

LP relaxation

- What if we relax the integrality constraints? E.g., instead of setting $z \in \{0, 1\}$, we allow $z \in [0, 1]$ (i.e., $0 \leq z \leq 1$).
- In this way the MILP is reduced to a standard LP, and can be solved easily.
- There are three possible outcomes:
  1. The LP is not feasible: then the MILP is not feasible either.
  2. The LP is feasible, and the optimal solution is such that it satisfies the integrality constraints: then the solution from the LP is the optimal solution for the MILP as well (!)
  3. The LP is feasible, but the optimal solution is not integral: how to recover a solution for the MILP?

Integral LP relaxation

- It turns out that some IPs always admit a LP relaxation with integral solutions: hence, these IPs are very easy to solve.
- Examples include the shortest path problem discussed in previous lectures.
- Other examples include problems in which the "A" matrix is totally unimodular (i.e., all the determinants of non-singular square submatrices are ± 1), and the "b" vector is integral.
  - The entries of a totally unimodular matrix are either 0 or ± 1.
  - The matrix in the shortest path problem is in fact totally unimodular. From the example in the shortest-paths Lecture:

$$A = \begin{bmatrix} 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & -1 \\ -1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

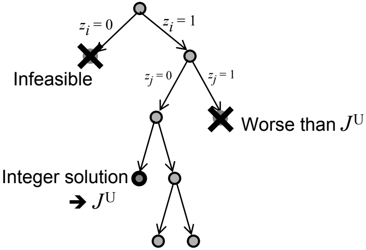
Branch and Bound

- In general, it is not always the case that the LP relaxation yields a valid solution.
- A very effective technique is based on **branch and bound** techniques
  - Branch into many subproblems, solve them using the LP relaxation (lower bound).
  - Keep track of the lower and upper bounds on the solutions found.
  - **Key idea**: if the lower bound on the subproblem is higher than the current upper bound om the original problem, then the subproblem does not need to be considered further.
  - Upper bound is given by a feasible solution. Lower bound is given by a relaxed problem where $z_i \in \{0, 1\}$ is replaced with $0 \leq z \leq 1$.

Branch and Bound Algorithm

1. Solve the LP relaxation of the MILP, call $\bar{J}$ the optimal cost.
2. If the LP solution is integral, terminate, $\bar{J}$ is optimal. If the LP is infeasible, terminate, the problem is not feasible. Else, set $J^U \leftarrow \infty$, $J^L \leftarrow \tilde{J}$.
3. Pick one of the z variables, and create two sub-problems setting this variable to 0 and then to 1.
4. For each of the subproblem, solve the LP relaxation, call $\bar{J}$ the optimal cost.
5. If the LP solution is integral, then it is a candidate optimal solution. Update $J^U \leftarrow \min\{J^U, \bar{J}\}$.
6. Else, if the LP solution is not integral, but $\bar{J} > J^U$, then there is no value in further exploring that subproblem. Prune the branch.
7. Else, if the LP is not integral, but $\bar{J} < J^U$, then continue branching: create other subproblems from this problem.
8. If the LP is infeasible, then prune the branch.

Branch and Bound Algorithm



- As in graph search, branch and bound may eliminate the need to explore all the possible choices for the integer variables.
- This and similar methods are at the basis of most state-of-the-art open-source and commercial solvers, e.g., GLPK, LP_SOLVE, and ILOG CPLEX.

Remarks on MILPs

- **Pros**:
  - Very general formulation: you can write a very large class of motion planning problems in this way.
  - The problems "look" like LPs.
  - Very powerful commercial solvers available: Ilogs CPLEX can solve many of these problems quickly.
  - In some cases, amenable to real-time implementation *(Prof. How and his student have demonstrated real-time MILP-based planning on UAVs)*
- **Cons**:
  - Too general formulation: any problem can be converted into a MILP (!)
  - With generality comes complexity: MILPS are NP-hard (i.e., require exponential time to solve, in the worst case).
  - The dimension of the MILP can grow very quickly with the number of time steps/obstacles/vehicles.
  - The number of "or" constraints, i.e., of integer variables, is the key complexity driver.
- In general:
  - Non-convex optimization problems are hard.
  - Approximation algorithms can come in handy: e.g., relaxations
  - In certain conditions, approximations actually provide the optimal solution!

16.410 / 16.413 Principles of Autonomy and Decision Making

Fall 2010