16.410/413
Principles of Autonomy and Decision Making

Lecture 16: Mathematical Programming I

Emilio Frazzoli

Aeronautics and Astronautics
Massachusetts Institute of Technology

November 8, 2010

Assignments

Readings

- Lecture notes
- [IOR] Chapters 2, 3, 9.1-3.
- [PA] Chapter 6.1-2

Shortest Path Problems on Graphs

Input: $\langle V, E, w, s, G \rangle$:

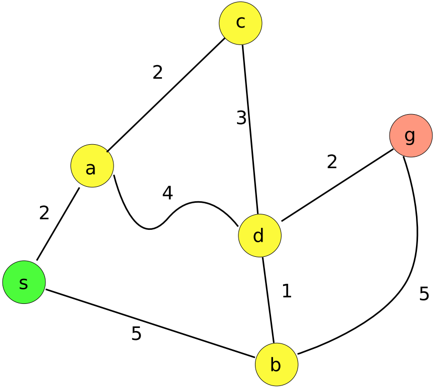
- V: set of vertices (finite, or in some cases countably infinite).
- $E \subseteq V \times V$: set of edges.
- $w : E \to \mathbb{R}_+, e \mapsto w(e)$: a function that associates to each edge a strictly positive weight (cost, length, time, fuel, prob. of detection).
- $S, G \subseteq V$: respectively, start and end sets. Either S or G, or both, contain only one element. For a point-to-point problem, both S and G contain only one element.

Output: $\langle T, W \rangle$

- T is a weighted tree (graph with no cycles) containing one minimum-weight path for each pair of start-goal vertices $(s, g) \in S \times G$.
- $W : S \times G \to \mathbb{R}_+$ is a function that returns, for each pair of start-goal vertices $(s, g) \in S \times G$, the weight $W(s, g)$ of the minimum-weight path from s to g. The weight of a path is the sum of the weights of its edges.

Example: point-to-point shortest path

Find the minimum-weight path from s to g in the graph below:



Solution: a simple path $P = \langle s, a, d, g \rangle$ ($P = \langle s, b, d, g \rangle$ would be acceptable, too), and its weight $W(s, g) = 8$.

Another look at shortest path problems

Cost formulation

- The cost of a path P is the sum of the cost of the edges on the path. *Can we express this as a simple mathematical formula?*
  - Label all the edges in the graph with consecutive integers, e.g., $E = \{e_1, e_2, \ldots, e_{n_E}\}$.
  - Define $w_i = w(e_i)$, for all $i \in 1, \ldots, n_E$.
  - Associate with each edge a variable x_i, such that:

$$x_i = \begin{cases} 1 & \text{if } e_i \in P, \\ 0 & \text{otherwise.} \end{cases}$$

- Then, the cost of a path can be written as:

$$\mathrm{Cost}(P) = \sum_{i=1}^{n_E} w_i x_i.$$

- Notice that the cost is a **linear function** of the unknowns $\{x_i\}$

Another look at shortest path problems (2)

Constraints formulation

- Clearly, if we just wanted to minimize the cost, we would choose $x_i = 0$, for all $i = 1, \ldots, n_E$: this would not be a path connecting the start and goal vertices (in fact, it is the empty path).
- Add these constraints:
  - There must be an edge in P that goes out of the start vertex.
  - There must be an edge in P that goes into the goal vertex.
  - Every (non start/goal) node with an incoming edge must have an outgoing edge
- A neater formulation is obtained by adding a “virtual” edge e_0 from the goal to the start vertex:
  - $x_0 = 1$, i.e., the virtual edge is always chosen.
  - Every node with an incoming edge must have an outgoing edge

Another look at shortest path problems (3)

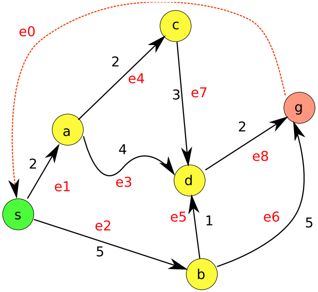
- Summarizing, what we want to do is:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{n_E} w_i x_i \\
\text{subject to:} \quad & \sum_{e_i \in \mathrm{In}(s)} x_i - \sum_{e_j \in \mathrm{Out}(s)} x_j = 0, \quad \forall s \in V; \\
& x_i \geq 0, \quad i = 1, \ldots, n_E; \\
& x_0 = 1.
\end{aligned}
$$

- It turns out that the solution of this problem yields the shortest path. (Interestingly, we do not have to set that $x_i \in \{0, 1\}$, this will be automatically satisfied by the optimal solution!)

Another look at shortest path problems (4)

Consider again the following shortest path problem:



$$
\begin{aligned}
\min \quad & 2x_1 + 5x_2 + 4x_3 + 2x_4 + x_5 + 5x_6 + 3x_7 + 2x_8 \\
\text{s.t.:} \quad & x_0 - x_1 - x_2 = 0, (\text{node } s); \\
& x_1 - x_3 - x_4 = 0, (\text{node } a); \\
& x_2 - x_5 - x_6 = 0, (\text{node } b); \\
& x_4 - x_7 = 0, (\text{node } c); \\
& x_3 + x_5 + x_7 - x_8 = 0, (\text{node } c); \\
& x_2 + x_5 - x_0 = 0, (\text{node } g); \\
& x_i \geq 0, \quad i = 1, \ldots, 8; \\
& x_0 = 1.
\end{aligned}
$$

Notice: cost function and constraints are affine ("linear") functions of the unknowns $(x_1, \ldots, x_8)$.

A fire-fighting problem: formulation

Three fires

- Fire 1 needs 1000 units of water;
- Fire 2 needs 2000 units of water;
- Fire 3 needs 3000 units of water.

Two fire-fighting autonomous aircraft

- Aircraft A can deliver 1 unit of water per unit time;
- Aircraft B can deliver 2 units of water per unit time.

Objective

It is desired to extinguish all the fires in minimum time.

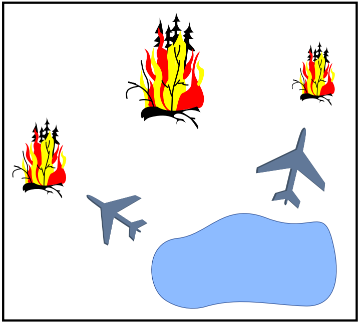


Image by MIT OpenCourseWare.

A fire-fighting problem: formulation (2)

- Let t_{A1}, t_{A2}, t_{A3} the the time vehicle A devotes to fire 1, 2, 3, respectively.
  Definte t_{B1}, t_{B2}, t_{B3} in a similar way, for vehicle B.
- Let T be the total time needed to extinguish all three fires.
- Optimal value (and optimal strategy) found solving the following problem:

$$
\begin{aligned}
\text{min} \quad & T \\
\text{s.t.:} \quad & t_{A1} + 2t_{B1} = 1000, \\
& t_{A2} + 2t_{B2} = 2000, \\
& t_{A3} + 2t_{B3} = 3000, \\
& t_{A1} + t_{A2} + t_{A3} \leq T, \\
& t_{B1} + t_{B2} + t_{B3} \leq T, \\
& t_{A1}, t_{A2}, t_{A3}, t_{B1}, t_{B2}, t_{B3}, T \geq 0.
\end{aligned}
$$

- (if you are curious about the solution, the optimal T is 2000 time units)

Outline

1. Mathematical Programming
2. Linear Programming
3. Geometric Interpretation

Mathematical Programming

- Many (most, maybe all?) problems in engineering can be defined as:
  - A set of constraints defining all candidate ("feasible") solutions, e.g., $g(x) \leq 0$.
  - A cost function defining the "quality" of a solution, e.g., $f(x)$.

- The formalization of a problem in these terms is called a **Mathematical Program**, or **Optimization Problem**.
  (Notice this has nothing to do with "computer programs!")

- The two problems we just discussed are examples of mathematical program. Furthermore, both of them are such that both f and g are affine functions of x. Such problems are called **Linear Programs**.

Outline

1. Mathematical Programming
2. Linear Programming
   - Historical notes
   - Geometric Interpretation
   - Reduction to standard form
3. Geometric Interpretation

Linear Programs

- The **Standard Form** of a linear program is an optimization problem of the form

$$
\begin{aligned}
\max \quad & z = c_1 x_1 + c_2 x_2 + \ldots, c_n x_n, \\
\text{s.t.:} \quad & a_{11} x_1 + a_{12} x_2 + \ldots + a_{1n} x_n = b_1, \\
& a_{21} x_1 + a_{22} x_2 + \ldots + a_{2n} x_n = b_2, \\
& \ldots \\
& a_{m1} x_1 + a_{m2} x_2 + \ldots + a_{mn} x_n = b_m, \\
& x_1, x_2, \ldots, x_n \geq 0.
\end{aligned}
$$

- In a more compact form, the above can be rewritten as:

$$
\begin{aligned}
\min \quad & z = c^T x, \\
\text{s.t.:} \quad & Ax = b, \\
& x \geq 0.
\end{aligned}
$$

Historical Notes

- Historical contributor: **G. Dantzig** (1914-2005), in the late 1940s. (He was at Stanford University.) Realize many real-world design problems can be formulated as linear programs and solved efficiently. Finds algorithm, the Simplex method, to solve LPs. As of 1997, still best algorithm for most applications.

Historical Notes

- Historical contributor: **G. Dantzig** (1914-2005), in the late 1940s. (He was at Stanford University.) Realize many real-world design problems can be formulated as linear programs and solved efficiently. Finds algorithm, the Simplex method, to solve LPs. As of 1997, still best algorithm for most applications.
- So important for world economy that any new algorithmic development on LPs is likely to make the front page of major newspapers (e.g. NY times, Wall Street Journal). Example: 1979 L. Khachyans adaptation of ellipsoid algorithm, N. Karmarkars new interior-point algorithm.

Historical Notes

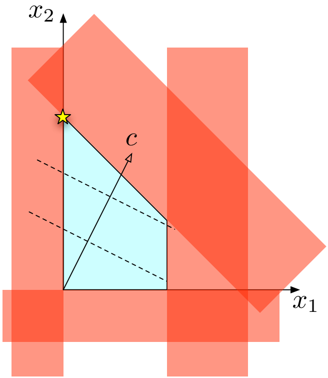
- Historical contributor: **G. Dantzig** (1914-2005), in the late 1940s. (He was at Stanford University.) Realize many real-world design problems can be formulated as linear programs and solved efficiently. Finds algorithm, the Simplex method, to solve LPs. As of 1997, still best algorithm for most applications.
- So important for world economy that any new algorithmic development on LPs is likely to make the front page of major newspapers (e.g. NY times, Wall Street Journal). Example: 1979 L. Khachyans adaptation of ellipsoid algorithm, N. Karmarkars new interior-point algorithm.
- A remarkably practical and theoretical framework: LPs eat a large chunk of total scientific computational power expended today. It is crucial for economic success of most distribution/transport industries and to manufacturing.

Historical Notes

- Historical contributor: **G. Dantzig** (1914-2005), in the late 1940s. (He was at Stanford University.) Realize many real-world design problems can be formulated as linear programs and solved efficiently. Finds algorithm, the Simplex method, to solve LPs. As of 1997, still best algorithm for most applications.
- So important for world economy that any new algorithmic development on LPs is likely to make the front page of major newspapers (e.g. NY times, Wall Street Journal). Example: 1979 L. Khachyans adaptation of ellipsoid algorithm, N. Karmarkars new interior-point algorithm.
- A remarkably practical and theoretical framework: LPs eat a large chunk of total scientific computational power expended today. It is crucial for economic success of most distribution/transport industries and to manufacturing.
- Now becomes suitable for real-time applications, often as the fundamental tool to solve or approximate much more complex optimization problem.

- Consider the following simple LP:

$$
\begin{aligned}
\text{max} \quad & z = x_1 + 2x_2 = (1, 2) \cdot (x_1, x_2), \\
\text{s.t.:} \quad & x_1 \le 3, \\
& x_1 + x_2 \le 5, \\
& x_1, x_2 \ge 0.
\end{aligned}
$$

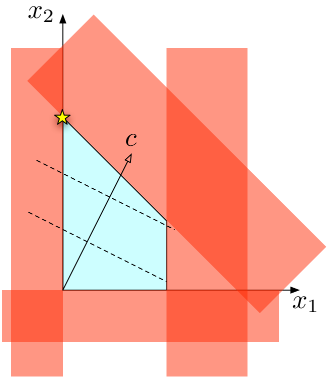


- Each inequality constraint defines a hyperplane, and a **feasible** half-space.
- The intersection of all feasible half spaces is called the **feasible region**.
- The feasible region is a (possibly **unbounded**) polyhedron.
- The feasible region could be the empty set: in such case the problem is said **unfeasible**.

Geometric Interpretation (2)

- Consider the following simple LP:

$$\begin{aligned} \max \quad & z = x_1 + 2x_2 = (1,2)\cdot(x_1,x_2), \\ \text{s.t.:} \quad & x_1 \le 3, \\ & x_1 + x_2 \le 5, \\ & x_1, x_2 \ge 0. \end{aligned}$$



- The "c" vector defines the gradient of the cost.
- Constant-cost loci are planes normal to c.
- Most often, the optimal point is located at a vertex (corner) of the feasible region.
  - If there is a single optimum, it must be a corner of the feasible region.
  - If there are more than one, two of them must be adjacent corners.
  - If a corner does not have any adjacent corner that provides a better solution, then that corner is in fact the optimum.

Converting a LP into standard form

- Convert to maximization problem by flipping the sign of c.
- Turn all "technological" inequality constraints into equalities:
  - **less than** constraints: introduce **slack** variables.

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i \Rightarrow \sum_{j=1}^{n} a_{ij}x_j + s_i = b_i, \quad s_i \geq 0.$$

  - **greater than** constraints: introduce **excess** variables.

$$\sum_{j=1}^{n} a_{ij}x_j \geq b_i \Rightarrow \sum_{j=1}^{n} a_{ij}x_j - e_i = b_i, \quad e_i \geq 0.$$

- Flip the sign of non-positive variables: $x_i \leq 0 \Rightarrow x_i' = -x_i \geq 0$.
- If a variable does not have sign constraints, use the following trick:

$$x_i \Rightarrow x_i' - x_i'', \quad x_i', x_i'' \geq 0.$$

Outline

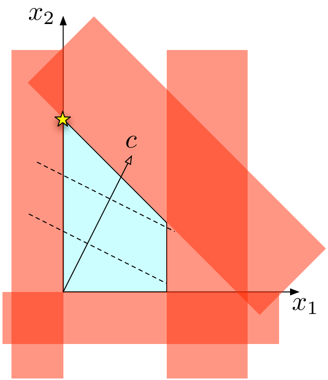
1. Mathematical Programming
2. Linear Programming
3. Geometric Interpretation
   - Geometric Interpretation

Geometric Interpretation

- Consider the following simple LP:

$$\begin{aligned} \max \quad & z = x_1 + 2x_2 = (1,2)\cdot(x_1,x_2), \\ \text{s.t.:} \quad & x_1 \le 3, \\ & x_1 + x_2 \le 5, \\ & x_1, x_2 \ge 0. \end{aligned}$$

- Each inequality constraint defines a hyperplane, and a **feasible** half-space.
- The intersection of all feasible half spaces is called the **feasible region**.



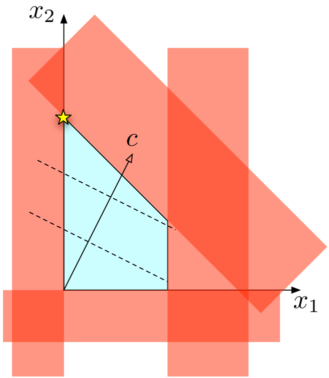
- The feasible region is a (possibly **unbounded**) polyhedron.
- The feasible region could be the empty set: in such case the problem is said **unfeasible**.

Geometric Interpretation (2)

- Consider the following simple LP:

$$\begin{aligned} \max \quad & z = x_1 + 2x_2 = (1,2)\cdot(x_1,x_2), \\ \text{s.t.:} \quad & x_1 \le 3, \\ & x_1 + x_2 \le 5, \\ & x_1, x_2 \ge 0. \end{aligned}$$

- The "c" vector defines the gradient of the cost.
- Constant-cost loci are planes normal to c.



- Most often, the optimal point is located at a vertex (corner) of the feasible region.
  - If there is a single optimum, it must be a corner of the feasible region.
  - If there are more than one, two of them must be adjacent corners.
  - If a corner does not have any adjacent corner that provides a better solution, then that corner is in fact the optimum.

A naïve algorithm (1)

- Recall the standard form:

$$\begin{aligned} \min \quad & z = c^T x \\ \text{s.t.:} \quad & Ax = b, \\ & x \geq 0. \end{aligned}$$

- Corners of the feasible regions (also called **basic feasible solutions**) are solutions of $Ax = b$ (m equations in n unknowns, $n > m$), obtained setting $n - m$ variables to zero, and solving for the others (basic variables), ensuring that all variables are non-negative.

A naïve algorithm (1)

- Recall the standard form:

$$\begin{array}{ll} \min & z = c^T x \\ \text{s.t.:} & Ax = b, \\ & x \geq 0. \end{array} \quad \left(\text{or, really:} \quad \begin{array}{ll} \min & z = c_y^T y + c_s^T s \\ \text{s.t.:} & A_y y + Is = b, \\ & y, s \geq 0. \end{array} \right)$$

- Corners of the feasible regions (also called **basic feasible solutions**) are solutions of $Ax = b$ (m equations in n unknowns, $n > m$), obtained setting $n - m$ variables to zero, and solving for the others (basic variables), ensuring that all variables are non-negative.
- This amounts to:
  - picking n_y inequality constraints, (notice that $n = n_y + n_s = n_y + m$).
  - making them active (or binding),
  - finding the (unique) point where all these hyperplanes meet.
  - If all the variables are non-negative, this point is in fact a vertex of the feasible region.

A naïve algorithm (2)

- One could possibly generate all basic feasible solutions, and then check the value of the cost function, finding the optimum by enumeration.

- **Problem**: how many candidates?

$$\binom{n}{n-m} = \frac{n!}{m!(n-m)!}.$$

  - for a "small" problem with $n = 10$, $m = 3$, we get 120 candidates.
  - this number grows very quickly, the typical size of realistic LPs is such that n,m are often in the range of several hundreds, or even thousands.

- Much more clever algorithms exist: stay tuned.

16.410 / 16.413 Principles of Autonomy and Decision Making

Fall 2010