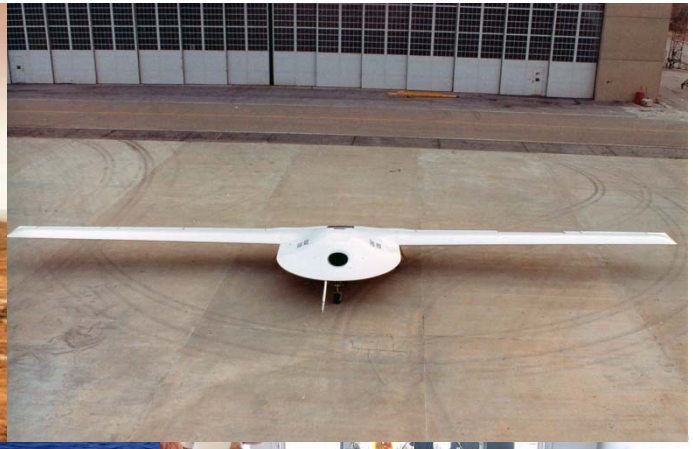
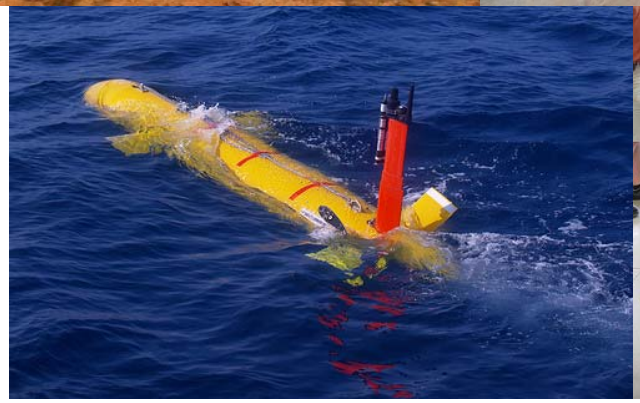



ME 597: Autonomous Mobile Robotics
Section 2 – Optimization

Prof. Steven Waslander

Outline

- Optimization Theory
  - Unconstrained optimization
  - Conditions for optimality
  - Convexity
  - Complexity
  - Constrained Optimization
  - Dynamic Programming

Unconstrained Optimization

- Given a function that maps a vector of variables to the reals

$$f : \mathbb{R}^n \rightarrow \mathbb{R}$$

- Find the minimum (or maximum) values of *f(x)*

$$\min_{x \in \mathbb{R}^n} \quad f(x)$$

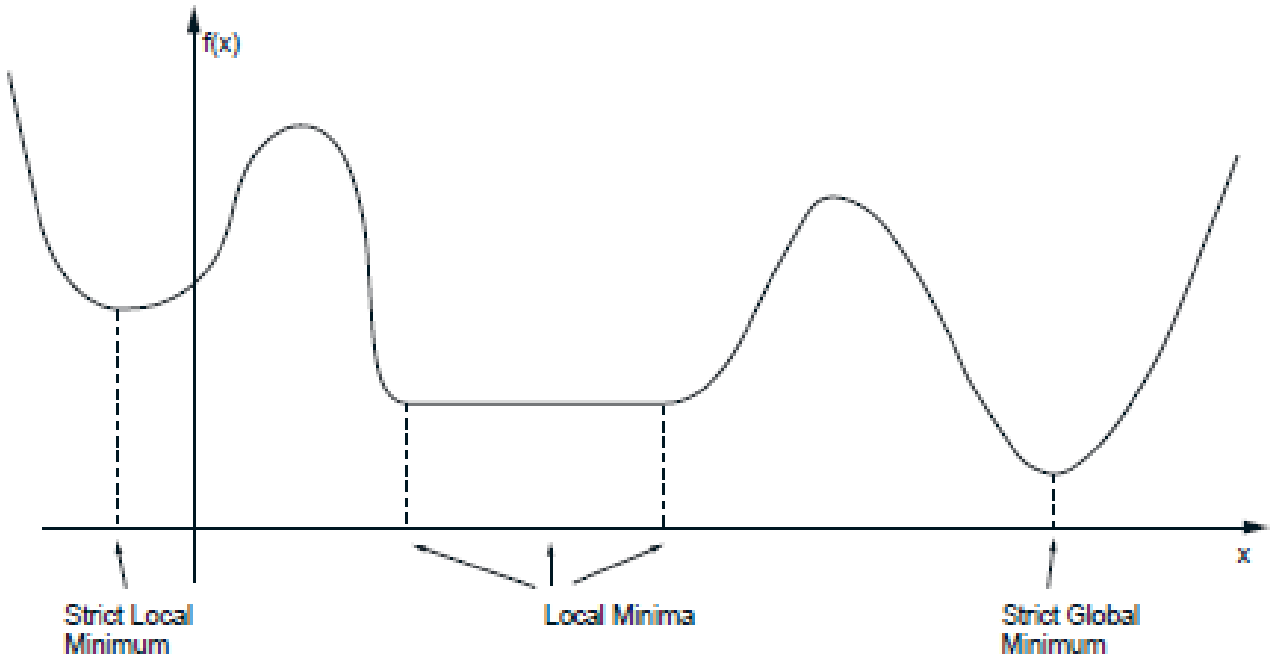
- Difficulty of problem depends on properties of *f*
  - Linear vs Nonlinear
  - Convex vs Nonconvex
  - Continuous vs Non-smooth vs Disjoint

Unconstrained Optimization

- Minima:
  - Local minimum x^*:

    There exists an $\varepsilon > 0$ such that

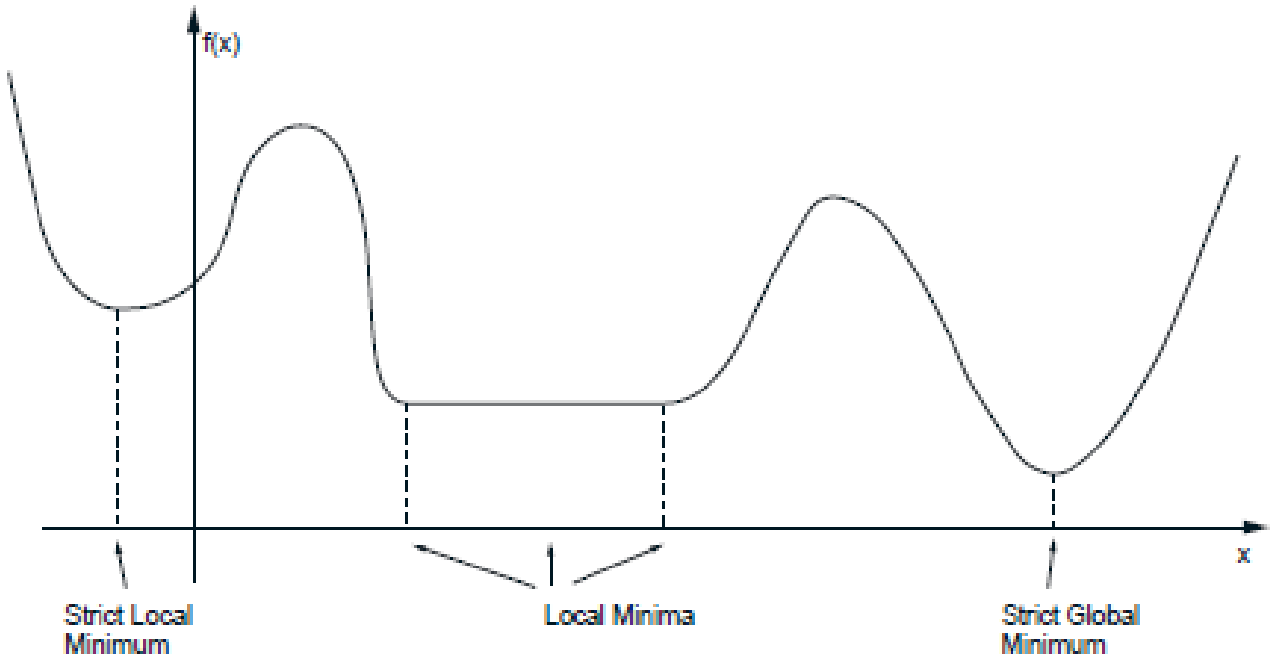
    $$f(x^*) \leq f(x), \text{ for all } x \text{ with } \| x - x^* \| < \varepsilon$$



Unconstrained Optimization

- Minima:
  - Global minimum:

$$f(x^*) \leq f(x), \text{ for all } x \in \Re^n$$

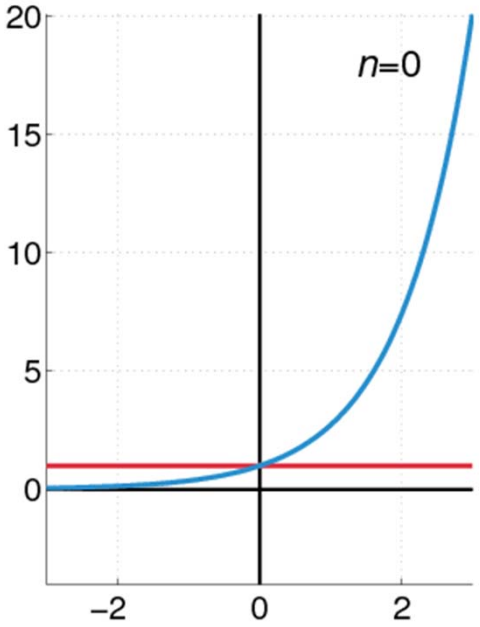


Conditions For Optimality

- For differentiable cost functions, can perform Taylor series expansion to find optimality conditions

  - Taylor series of *f(x)* about *x*

$$f(x+\Delta x) = f(x) + \nabla f(x)^T \Delta x + \tfrac{1}{2}\Delta x^T \nabla^2 f(x)\Delta x + H.O.T.$$



Courtesy of Wikipedia

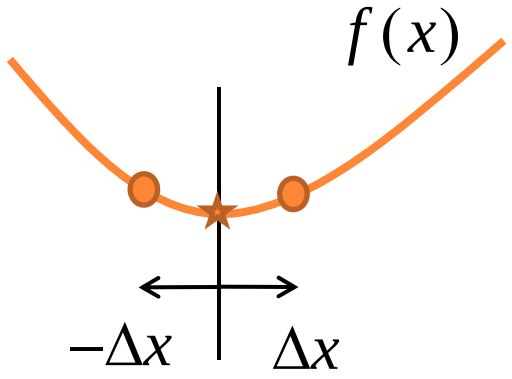
Conditions For Optimality

- Necessary conditions (NC)
  - If x^* is a local minimum, difference between minimum and nearby point should be non-negative by definition

$$f(x^* + \Delta x) - f(x^*) \geq 0$$

  - Similarly, for a negative step in x, the difference should be non-negative

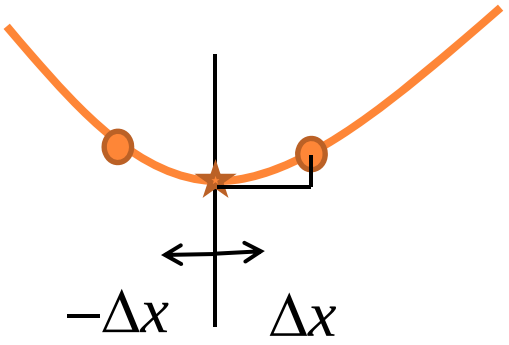
$$f(x^* - \Delta x) - f(x^*) \geq 0$$



Conditions For Optimality

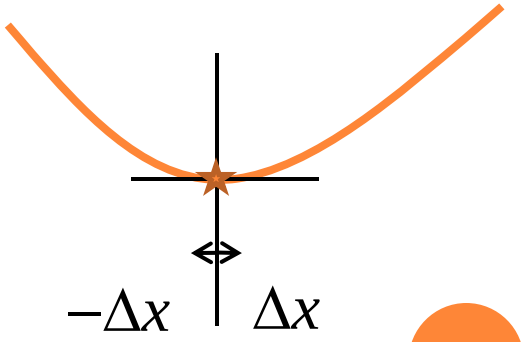
- Necessary conditions (NC)
  - As $\Delta x \to 0$, higher order terms in Taylor series disappear

$$f(x+\Delta x) - f(x) \approx \nabla f(x)^T \Delta x$$



  - First order term must satisfy above for Δx AND $-\Delta x$ in each element of x

$$\nabla f(x^*)^T \Delta x \geq 0 \text{ and } \nabla f(x^*)^T \Delta x \leq 0$$



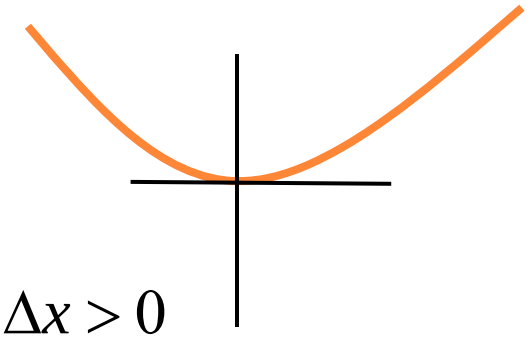
  - Necessary condition for optimality

$$\nabla f(x^*)^T = 0$$

Conditions for Optimality

- Sufficient conditions
  - Of all points that satisfy necessary conditions for optimality, which ones are truly local minima?
  - For all small excursions from optimal solution, cost increases
    - Since $\nabla f(x^*)^T = 0$
    - This means

$$f(x+\Delta x) - f(x) \approx \tfrac{1}{2}\Delta x^T \nabla^2 f(x)\Delta x > 0$$



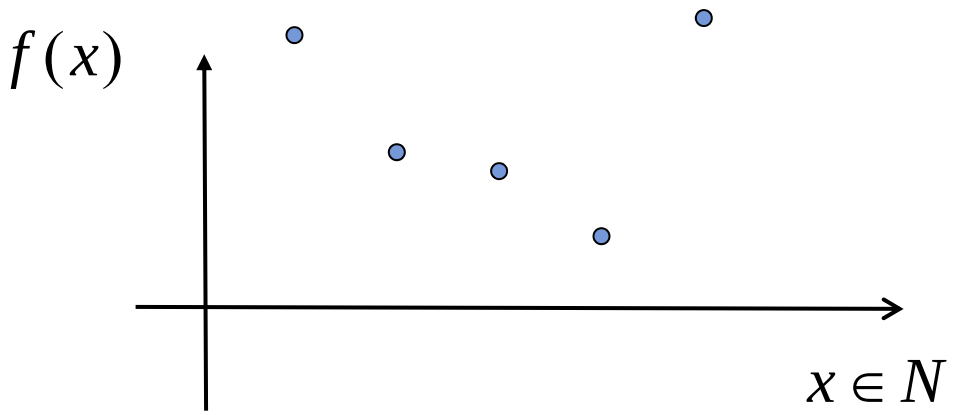
    - And so the sufficient condition for x^* to be a local minimum is

$$\nabla^2 f(x^*) \text{ is positive definite}$$

Optimality Conditions

- Note that these conditions are only useful if the gradient and Hessian exist

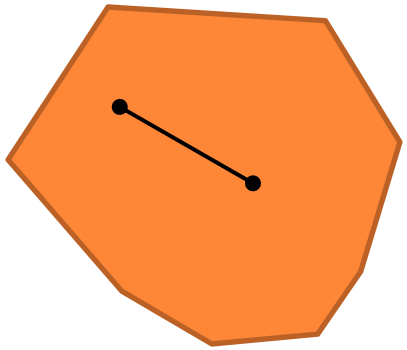
- Otherwise, resort to initial definition of optimality and demonstrate directly
  - Integer optimization



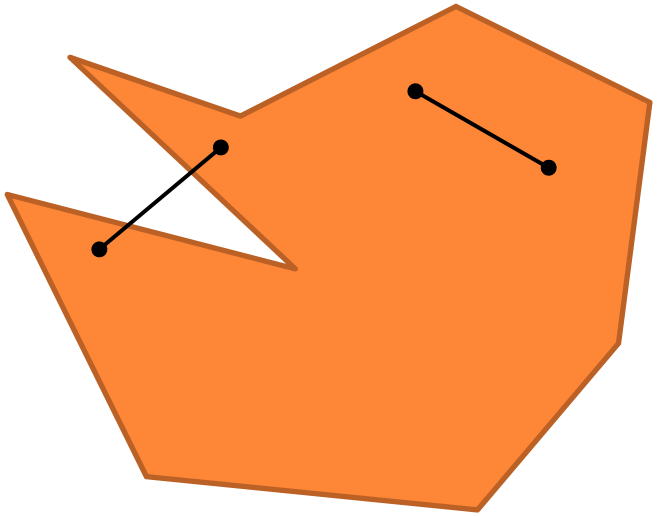
Convexity

- Definition: A set, C, is convex if any two points, x_1, x_2, in C can be connected by a line entirely in C.
  - That is, for all Θ in [0,1], we have

$$\theta x_1 + (1-\theta)x_2 \in C$$



Convex

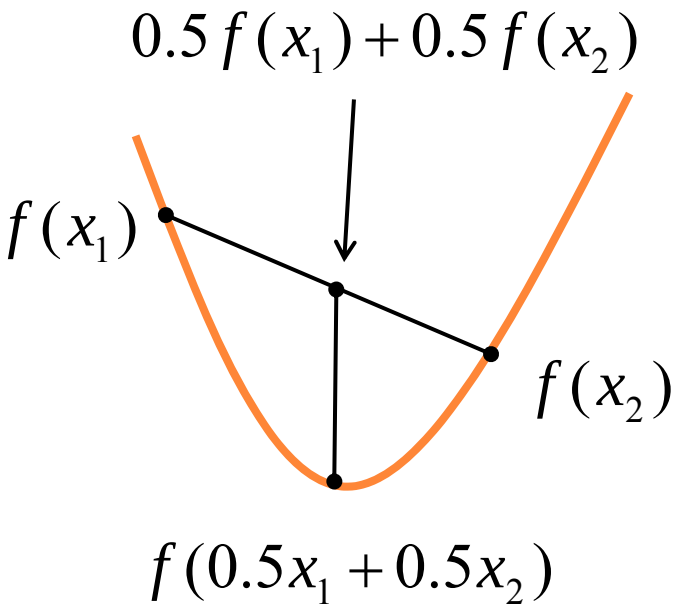


Nonconvex

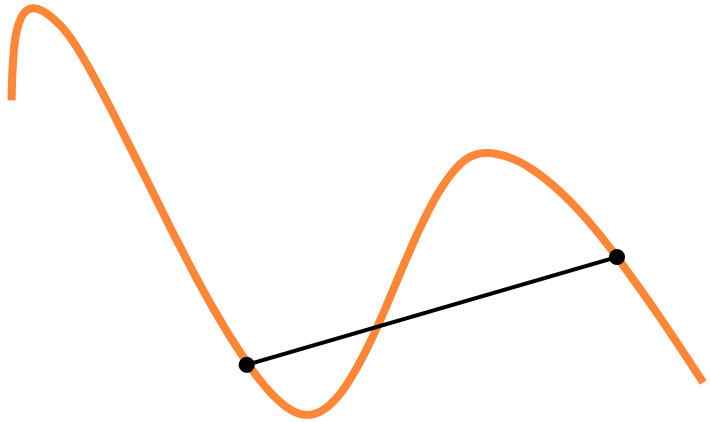
Convexity

- Definition: A function, f(x), is convex if for any two points, x_1, x_2, and for all Θ in [0,1], we have

$$f(\theta x_1 + (1-\theta)x_2) \le \theta f(x_1) + (1-\theta) f(x_2)$$



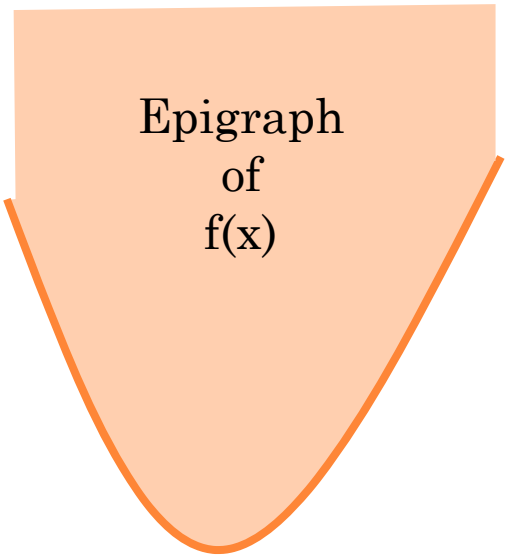
Convex

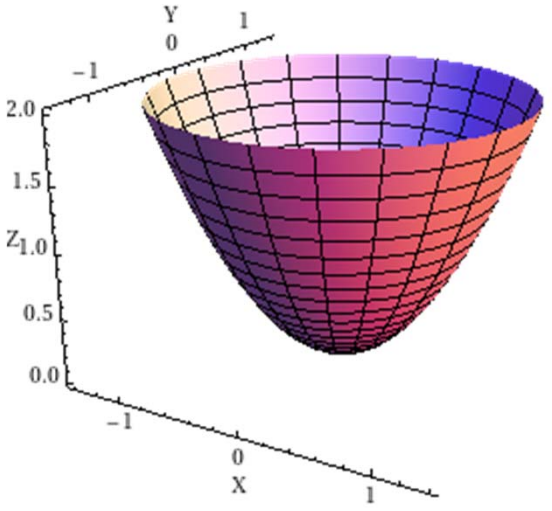


Nonconvex

Convexity

- A convex function has an epigraph that is a convex set
- Definition: A Convex Optimization problem is one where
  - f(x) is a convex function
  - g(x) is a convex function
  - h(x) is an affine function
- This definition ensure the feasible region is a convex set
- Convex optimization problems have a unique global minimum!





Complexity Analysis

- **(P)** – Deterministic **P**olynomial time algorithm
- **(NP)** – **N**on-deterministic **P**olynomial time algorithm,
  - Feasibility can be determined in polynomial time
- **(NP-complete)** – **NP** and at least as hard as any known **NP** problem
- **(NP-hard)** – not provably **NP** and at least as hard as any **NP** problem,
  - Optimization over an **NP-complete** feasibility problem

Constrained Optimization

- Standard form:

$$\begin{aligned} \min_{x \in X} \quad & f(x) \\ \text{subject to} \quad & g(x) \le 0 \\ & h(x) = 0 \end{aligned}$$

  - where

$$X \text{ can be any type of set}$$

$$f,\ g,\ h : X \to \mathbb{R}$$

- Specific classes of problems, depending on definitions of X, f, g, h.
- Very specific optimization engines, for every shade of problem

Optimization Problem Types

- Linear Program (LP)
  - (P) Easy, fast to solve, convex

$$\min_{x \in X \subseteq \Re^n} \quad f^T x$$
$$\text{s.t.} \quad Ax \le b$$
$$A_{eq} x = b_{eq}$$

- Matlab command:

```
x = linprog(f, A, b, Aeq, beq, LB, UB, x0)
```

- "How long do you think it would take to solve a problem with 1 million variables?"… "One second!"
  - Stephen Boyd, Stanford

Optimization Problem Types

- Quadratic Program (QP)
  - (P) Quadratic cost with linear constraints $O(n^3)$
    - Still fairly easy, fast to solve and convex

$$\min_{x \in X \subseteq \Re^n} \quad x^T Q x$$
$$\text{s.t.} \quad Ax \leq b$$
$$A_{eq} x = b_{eq}$$

- Matlab command:

```
x = quadprog(Q, A, b, Aeq, beq, LB, UB, x0)
```

- Kalman filter, LQR (unconstrained)
- In fact, any convex problem can be solved quickly
  - Matlab toolbox: cvx

Optimization Problem Types

- Non-Linear Program (NLP)
  - (P) Convex problems are easy to solve
  - Non-convex problems harder, not guaranteed to find global optimum (local minima can occur)

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & g(x) \leq 0 \\ & h(x) = 0 \\ \text{where,} \quad & f, g, h : \mathbb{R}^n \rightarrow \mathbb{R} \end{aligned}$$

Optimization Problem Types

- Mixed Integer Linear Program (MILP)
  - (NP-hard) computational complexity

$$\begin{aligned} \min_{x \in X} \quad & f^T x \\ \text{s.t.} \quad & Ax \le b \\ & A_{eq} = b_{eq} \\ \text{where } X \subseteq & Z^{n_i} \times \Re^{n_r} \end{aligned}$$

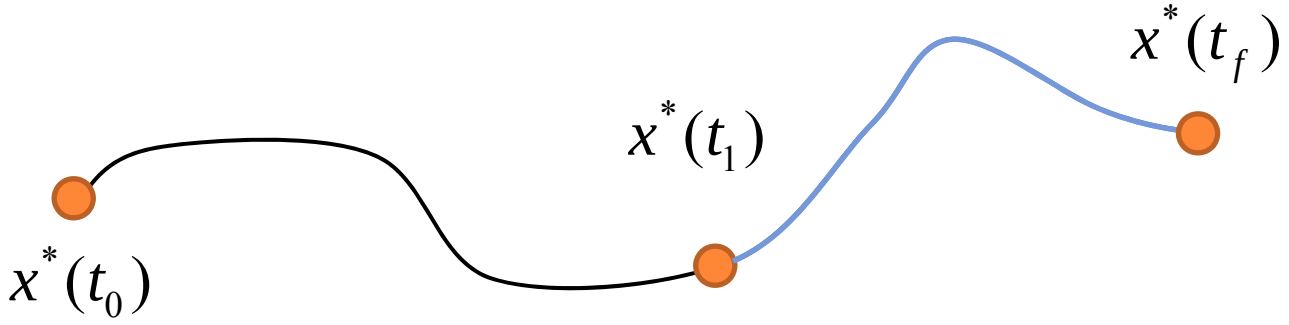
  - Exponential growth in complexity
  - However, many problems can be solved surprisingly quickly

- MINLP, MILQP etc.

Dynamic Programming

- Dynamic Programming
  - Richard Bellman (1953): Principle of Optimality
    - Applies to multi-period optimization problems
    - Discrete problems sum costs at each time step
    - Continuous problem costs are an integral over time interval

If a solution is optimal for periods t_0 to t_f, then the solution over any subinterval t_1 to t_f ($t_0 <= t_1 <= t_f$) must also be optimal



Dynamic Programming

- Discrete time case
  - In DP, state is state, inputs are actions
  - The sequence of all actions is a policy
  - Bellman Equation
    - Cost is written as a sum of stage costs

$$J_{t_0}(x_{t_0:t_f}) = \min \sum_{t=t_0}^{t_f} L_t(x_t)$$

    - Expressing the principle of optimality

$$J_t = \min_{x_t} \left[L_t(x_t) + J_{t+1} \right]$$

    - J_{t+1} is the "cost-to-go"

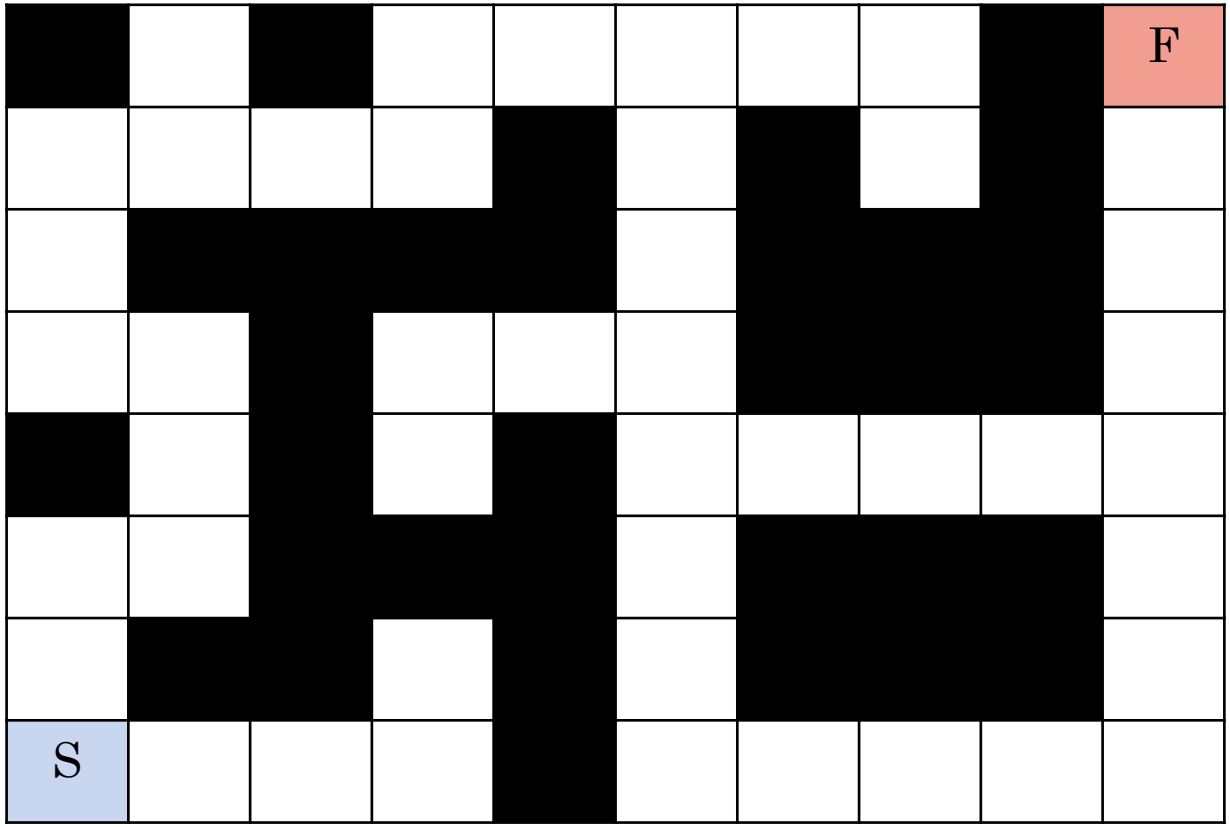
Dynamic Programming

- Can build optimal solutions by working through smaller sub-problems

- Discrete time, discrete space methods
  - Bottom-up
    - Solve trivial final stage problem first, then solve one step backward at a time
    - Results in a complete solution to every possible initial state
  - Top-down
    - Define a recursive program to solve sub-problems from a specific starting point
    - Sub-problem solutions are recorded and not re-solved
    - Results in a complete solution to every possible end state

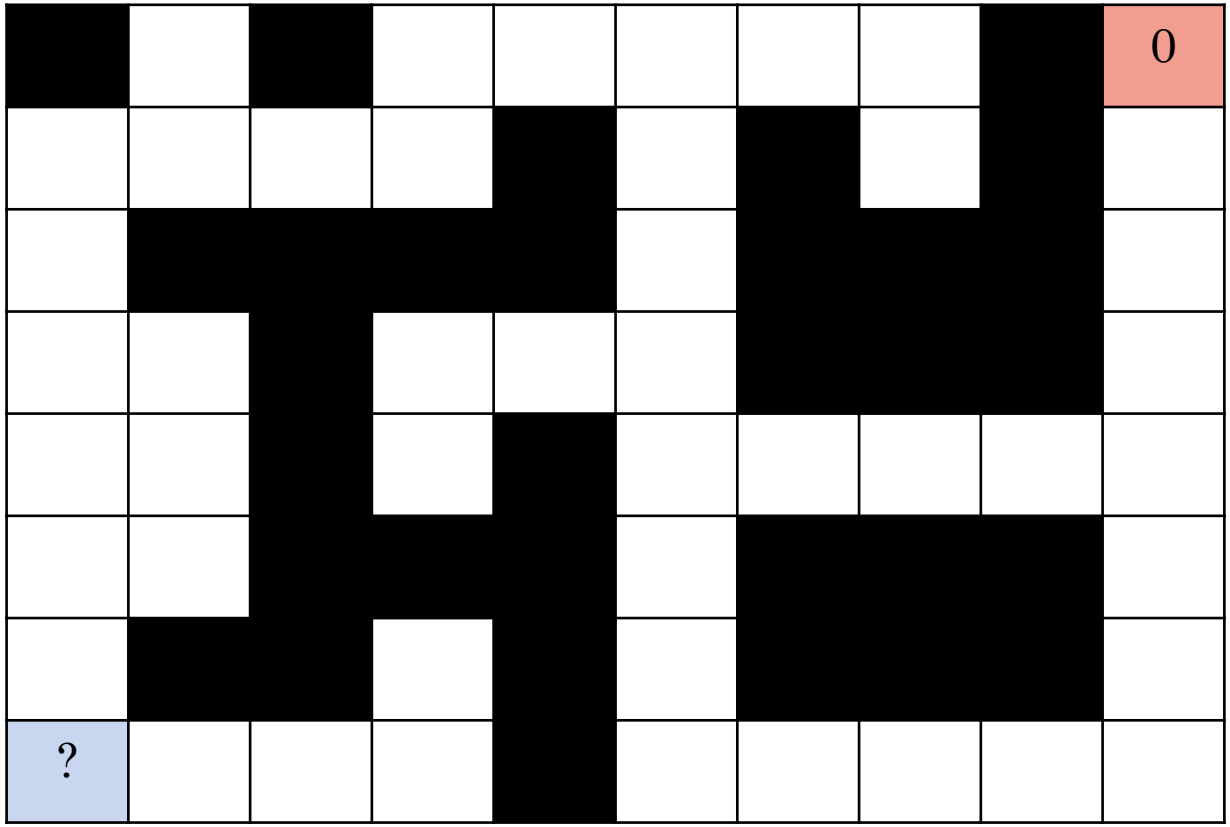
Example

- Maze: Discrete in time and space
  - S = Start, F = Finish



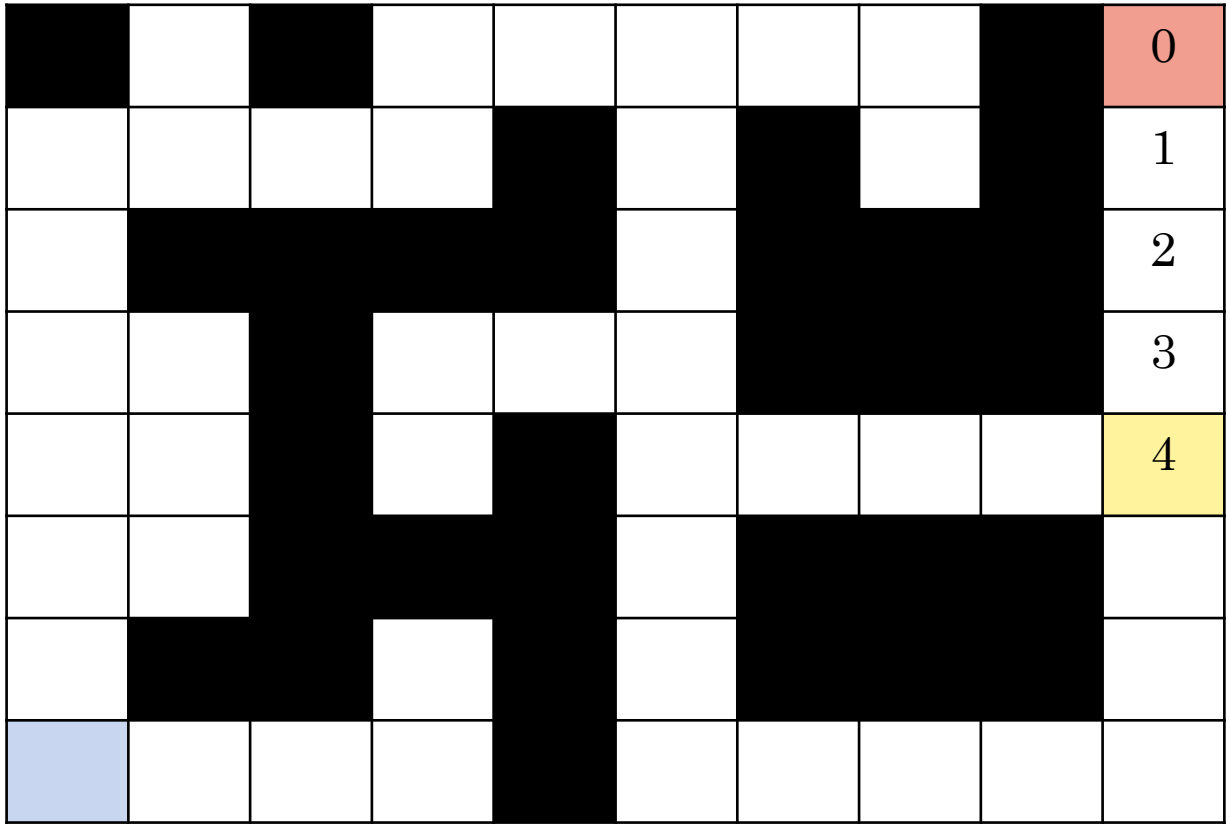
EXAMPLE

- Discrete Maze
  - $J_{tF} = 0$, Actions: Left, Up, Right, Down



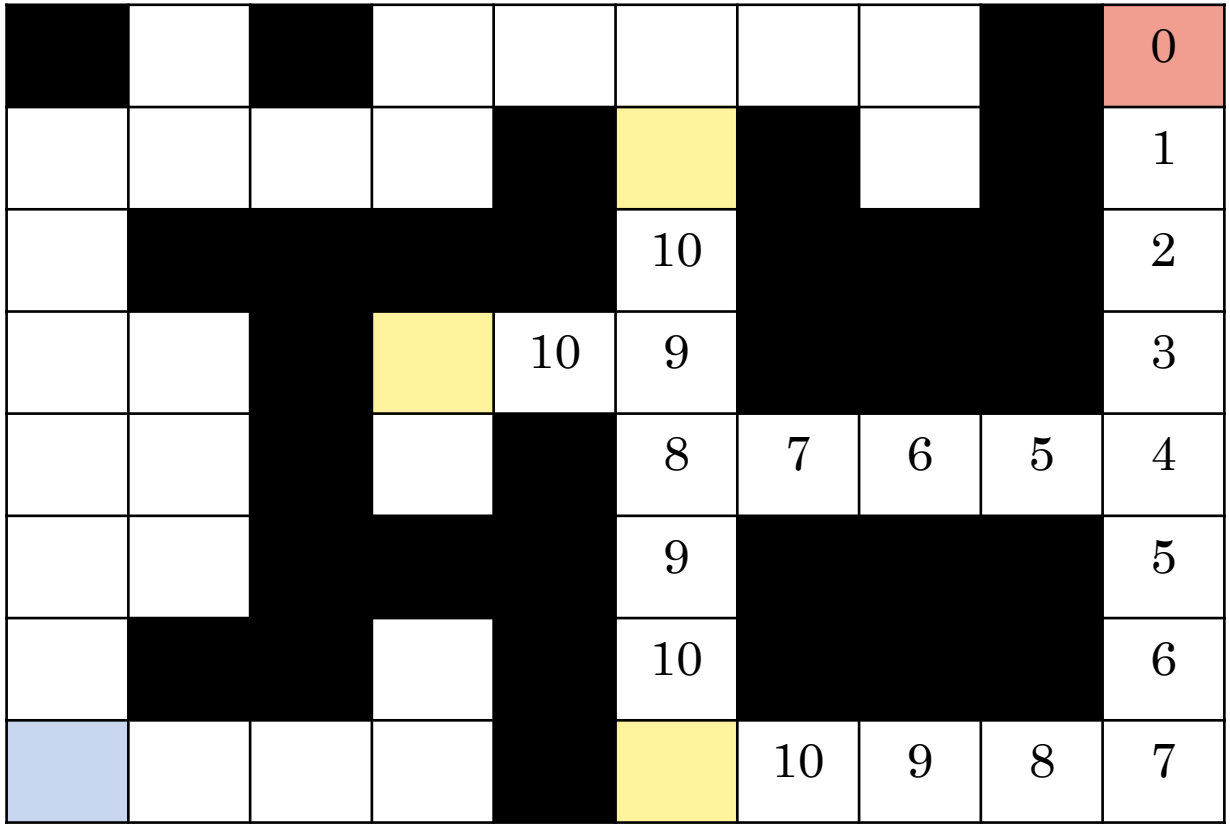
EXAMPLE

- Discrete Maze
  - Stage Cost $L_t=1$, step backward in time, filling in cost to go at each cell that can be reached



EXAMPLE

- Discrete Maze
  - Continuing …, bottom yellow cell has two options
  - $J_t = \min(L_t + J_{t+1}) = \min(1+10,1+10) = 11$



Example

- Discrete Maze
  - Continuing

	18		14	13	12	13	14		0
18	17	16	15		11		15		1
19					10				2
20	21		11	10	9				3
21			12		8	7	6	5	4
					9				5
					10				6
					11	10	9	8	7

Example

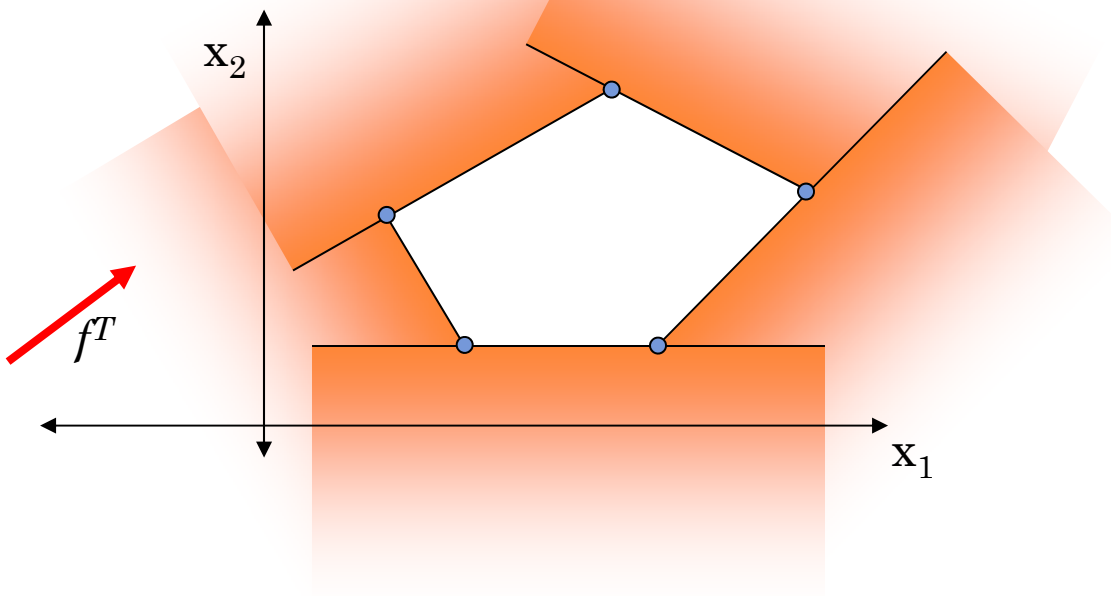
- Discrete Maze
  - So the cost from start to finish is 24

	18		14	13	12	13	14		0
18	17	16	15		11		15		1
19					10				2
20	21		11	10	9				3
21	22		12		8	7	6	5	4
22	23				9				5
23			28		10				6
24	25	26	27		11	10	9	8	7

Extra Slides

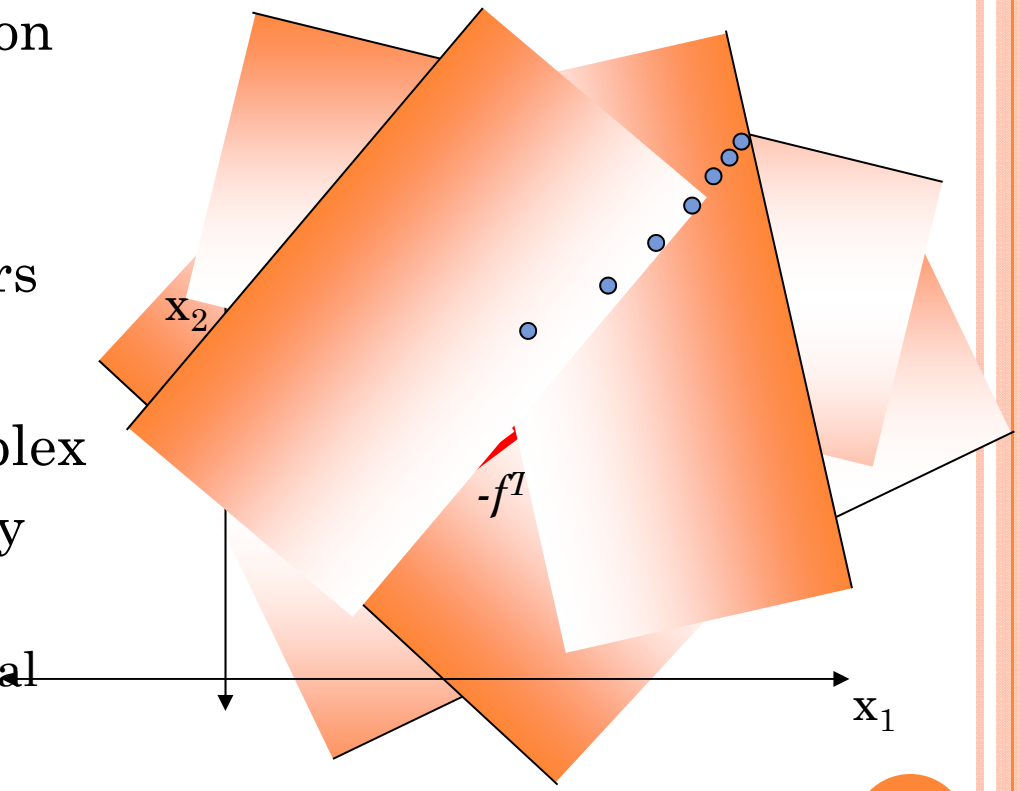
Solution Methods for Linear Programs

- Simplex Method
  - Optimum must be at the intersection of constraints
  - Intersections are easy to find, change inequalities to equalities, add slack variables
  - Jump from one vertex to the next (in a smart way), until no more improvement is possible



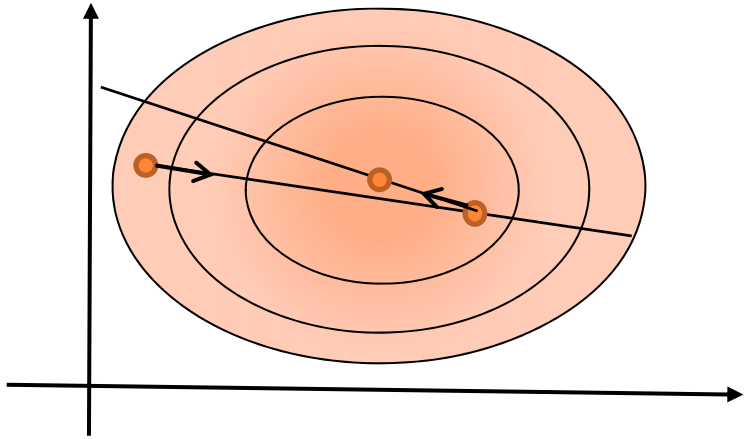
Solution Method for Linear Programs

- Interior Point Methods
  - Apply Barrier Function to each constraint and sum
  - Primal-Dual Formulation
  - Newton Step
  - At each iteration, increase slope of barriers
- Benefits
  - Scales better than Simplex
  - Certificate of Optimality
    - Stop whenever
    - Know how close to optimal the current solution is
    - Relies on duality



Solution Methods For NLPs

- Sequential Quadratic Programming
  - Also an interior point method
  - At each iteration, calculate gradient and Hessian of Lagrangian
  - If problem is a quadratic program, apply Newton step to optimal solution
  - If not, use Newton step direction as a descent direction and apply a line search
  - Finding Newton step involves inverse of Hessian



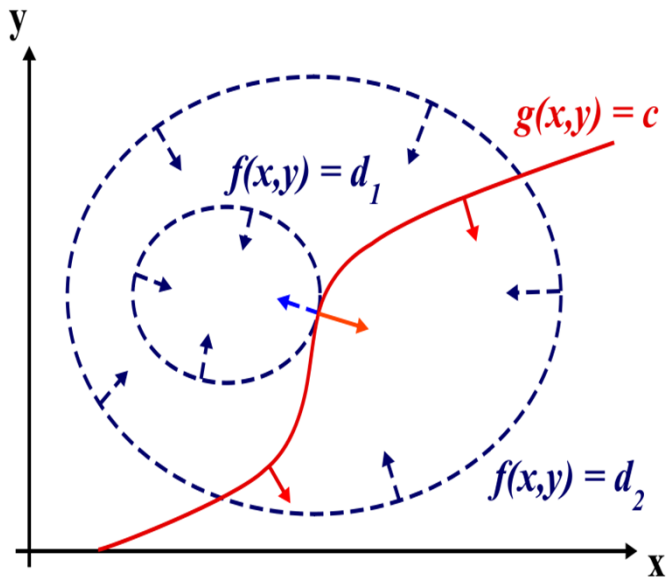
Solution Methods for MILPs

- Branch and Bound Algorithm
  1. Solve LP relaxation for lower bound on cost for current branch
     - If solution exceeds upper bound, branch is terminated
     - If solution is integer, replace upper bound on cost
  2. Create two branched problems by adding constraints to original problem
     - Select integer variable with fractional LP solution
     - Add integer constraints to the original LP
  3. Repeat until no branches remain, return optimal solution.

More details later!

Constrained Optimization

- Constrained minima
  - No active constraints = unconstrained
  - Active constraints
    - Gradient of cost must be perpendicular to active constraint
      - Otherwise moving along constraint would reduce cost and remain feasible
    - Can be expressed as $\nabla f(x^*, y^*) + \lambda^T \nabla g(x^*, y^*) = 0$
      - with Lagrange multiplier λ



Constrained Optimization

- Lagrange Multipliers
  - By introducing Lagrange multipliers, can convert constrained problem to an unconstrained problem
  - Can directly apply unconstrained optimization technique to Lagrangian

$$L(x) = f(x) + \lambda^T g(x) + \mu^T h(x)$$

  - Results in expanded necessary and sufficient conditions for optimality
  - In practice, best optimization algorithms treat constraints differently

Constrained Optimization

- Equality Constraints
  - Must be active

- Inequality Constraints (Karush, Kuhn, Tucker)
  - If active, Lagrange multipliers are non-negative
  - If inactive, Lagrange multipliers are zero

$$\nabla_x L(x^*, \lambda^*, \mu^*) = 0$$

$$\lambda_i^* \geq 0,\ i = 1, \ldots, m$$

$$\lambda_i^* = 0,\ i \notin A(x^*)$$

  - $A(x^*)$ is the set of active constraints