Overview of Optimization Techniques

Paulo C. P. Carvalho

IMPA - Instituto de Matemática Pura e Aplicada

Outline

- General concepts
  - Problem classification
  - Discretization: a user's choice
- Continuous optimization
  - Optimality conditions
  - Algorithms: basic ideas
- Combinatorial optimization
  - Algorithmic paradigms
  - Dynamic programming

Optimization

- Choosing the *best* among a set of *alternatives*

$$\min_{x \in S} \ f(x)$$

$f: S \to R$ is the *objective function*

S is the set of *feasible solutions*

Problem classification...

- According to the nature of S
  - variational
  - continuous
  - discrete
  - combinatorial

Problem classification ...

- Continuous optimization

  S has *finite* dimension, i.e., $S \subset R^n$
  (elements of S are *vectors*)

- Variational optimization

  S has *infinite* dimension
  (elements of S are *functions*)

Problem classification...

- Discrete optimization
  - usually, $S \subset Z^n$
- Combinatorial optimization
  - concise problem description by exploiting combinatorial structure
- Example: Traveling Salesman Problem

Traveling Salesman Problem

- Find the shortest tour connecting a set of cities (e.g., the capitals of the 48 contiguous states)

Traveling Salesman Problem

- Find the shortest tour connecting a set of cities (e.g., the capitals of the 48 contiguous states)

Traveling Salesman Problem

- Explicit description
  - each solution described by a 0-1 vector of length 2256 (number of directed arcs in the graph)
  - constraints:
    - for each city,
      number of incoming arcs =1
      number of outgoing arcs =1
    - subtour elimination constraints

Traveling Salesman Problem

- Combinatorial description
  - 48×48 matrix of distances
- Each possible solution is a circular permutation of the 48 cities
  - $47! \approx 10^{59}$ possibilities
  - can't be done by brute force, even today…
  - … but it was first solved in 1954, by exploiting combinatorial structure

Graphics: what type of optimization problem?

- Often, the type of optimization problem to be solved is a matter of the user's choice.
- Basic question: at which level to discretize?

Example

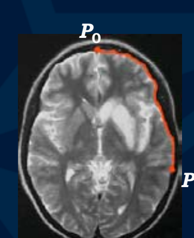
- Contour extraction

Find x: $[0, 1] \to R^2$, with $x(0) = P_0$ and $x(1) = P_1$, as to minimize

$$\int_0^1 \|x'(t)\| \, w(x(t)) \, dt,$$

where

$$w(x) = \frac{1}{k + \|\nabla I(x)\|}.$$



Possible approaches…

- Tackling directly the variational problem
  - infinitely-dimensional space
  - Euler-Lagrange formulation (partial differential equations)
  - solution process requires discretization

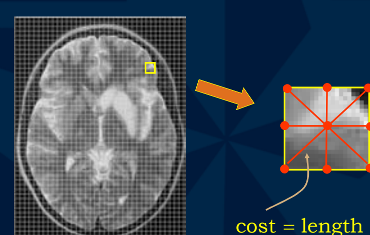
Possible approaches…

- Approximating by an optimization problem in finite dimension
  - for instance, restrict attention to cubic splines (characterized by a finite set of parameters)
  - use methods of continuous optimization in finite dimensions

Possible approaches…

- Approximating by a discrete optimization problem
  - discretization occurs at the beginning of the process
  - for instance, consider a regular grid and solve a shortest path problem

Possible approaches…



$$\text{cost} = \text{length} \cdot \frac{1}{k + \|\nabla I\|}$$

The need for discretization

- Discretization is always needed…
- … but can occur at different points of the solution process…
- … leading to different computational schemes

Outline

- General concepts
  - Problem classification
  - Discretization: a user's choice
- Continuous optimization
  - Optimality conditions
  - Algorithms: basic ideas
- Combinatorial optimization
  - Algorithmic paradigms
  - Dynamic programming

Optimality conditions

- Provide basis for solution methods
- Generalize classical theorems of differential calculus for general constraints

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to} \quad & h_i(x) = 0, \quad i = 1, \ldots, m \\ & g_j(x) \le 0, \quad j = 1, \ldots, p \end{aligned}$$

Necessary conditions for local optimality

- A (regular) local minimizer x_0 must satisfy the Karush-Kuhn-Tucker conditions

$$\nabla f(x_0) + \sum_{i=1}^{n} \lambda_i \nabla h_i(x_0) + \sum_{j=1}^{p} \mu_j \nabla g_j(x_0) = 0$$

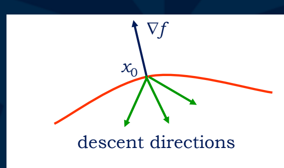
$$\sum_{j=1}^{p} \mu_j \; g_j(x_0) = 0$$

for some $\lambda \in R^m$, $\mu \in R^p$, $\mu \ge 0$

Special case: unconstrained problems

$$\nabla f(x_0) = 0$$

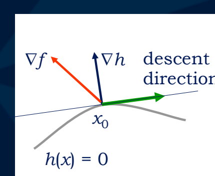
- otherwise, there is a descent direction.



Special case: equality constrained problems

$$\nabla f(x_0) = -\sum_{i=1}^{n} \lambda_i \nabla h_i(x_0)$$

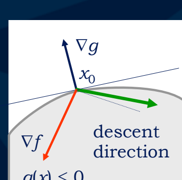
- the projection of ∇f on the tangent space must be zero; otherwise, there is a descent direction



Special case: inequality constrained problems

$$\nabla f(x_0) = -\sum_{\text{active } j} \mu_j \nabla g_j(x_0), \quad \mu_j \ge 0$$

- ∇f must be in the *cone* spanned by the negative gradients of active constraints; otherwise, there is a descent direction



Second order conditions

- Unconstrained case
  - conditions on the Hessian matrix $H = \nabla^2 f(x_0)$
  - necessary condition: $x^T H x \ge 0, \forall x$ (H is nonnegative)
  - sufficient condition: $x^T H x > 0, \forall x \ne 0$ (H is positive)
- Constrained case
  - Similar conditions, but with the operator ∇^2 restricted to the tangent space of active constraints

Solution methods

- Look for points that satisfy optimality conditions
- Explicit solution available only in very special cases (e.g., quadratic objective function, with linear equality constraints)
- Otherwise, have to resort to iterative methods

Example: least squares

$\min \| Ax - b \|$,

which is equivalent to

$\min \| Ax - b \|^2 = x^T(A^TA)x - 2x^TA^Tb + b^Tb$

(quadratic objective function)

Explicit solution:

$(A^TA)x - A^Tb = 0$

Iterative methods

- Two main paradigms
  - move along a descent direction
  - use linear approximation for ∇f to solve $\nabla f(x) = 0$ (*Newton's method*)
- Successful methods combine both strategies
  - e.g., Levenberg-Marquardt

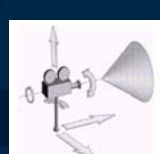
Main difficulties

- Global optimization is hard: conditions only for local optima, unless extra conditions hold (e.g., convexity).
- Iterative methods need good starting points.

Example: camera calibration...

- Unknown camera transformation T_θ

$$\begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{r}_1 & t_x \\ \mathbf{r}_2 & t_y \\ \mathbf{r}_3 & t_z \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = w \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$



- Parameters: f, t_x, t_y, t_z and the ortonormal basis $[\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3]$

Example: camera calibration...

- Given pairs $p_i \in \mathbf{R}^3$, $u_i \in \mathbf{R}^2$ $(i = 1, \ldots, n)$

$$\min_\theta \sum_{i=1}^{n} \| T_\theta(p_i) - u_i \|^2$$

- Initial solution by Tsai's method...
- ...then, Levenberg-Marquardt

Outline

- General concepts
  - Problem classification
  - Discretization: a user's choice
- Continuous optimization
  - Optimality conditions
  - Algorithms: basic ideas
- Combinatorial optimization
  - Algorithmic paradigms
  - Dynamic programming

Combinatorial optimization

- In general, simple to describe and hard to solve.
  - less mathematical structure than in continuous problems (calculus tools)
  - trivial solution by exhaustive search is not feasible (number of possible solutions usually grows exponentially with problem size)
  - class of NP-hard problems

Combinatorial optimization approaches

- Usually very dependent on the specific problem (*ad-hoc* methods)
- But there are a few general strategies, based on recursion
  - divide-and-conquer
  - dynamic programming (sequential optimization)

Dynamic Programming...

- Useful for problems that can be seen as a multi-stage decision process
- Principle of optimality
  - If an intermediate state is visited in an optimal sequence of decisions, then the decisions leading to that state must also be optimal.

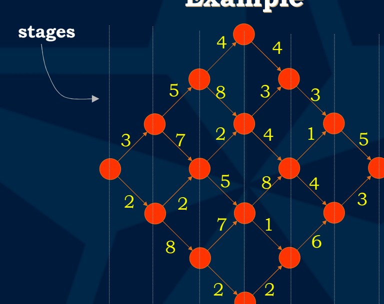
Dynamic Programming...

- Bellman's equation

$$f_{n+1}(x) = \min_{u \in \text{stage}\, n} \{f_n(u) + c(u,x)\}$$

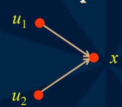
(to reach optimally *state x* in *stage n*+1 it suffices to consider those policies that reach optimally each possible *state u* in previous *stage n*)

Example



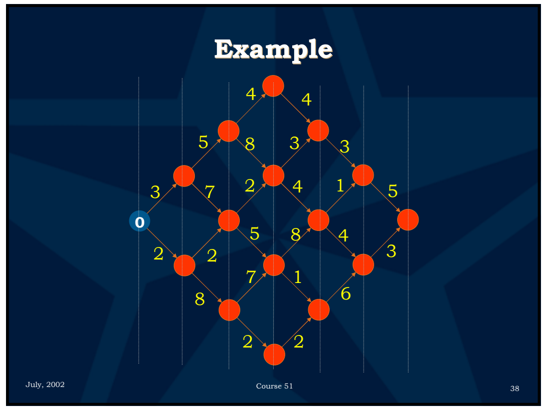
Solution approaches

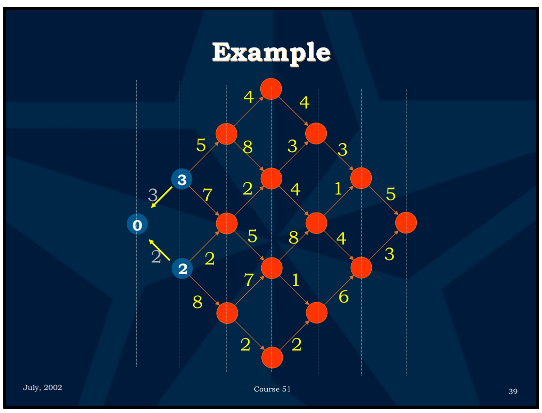
- $\binom{2n}{n} = \frac{(2n)!}{n!n!} \approx \frac{4^n}{\sqrt{2\pi n}}$ possible paths
- Solving Bellman's equations

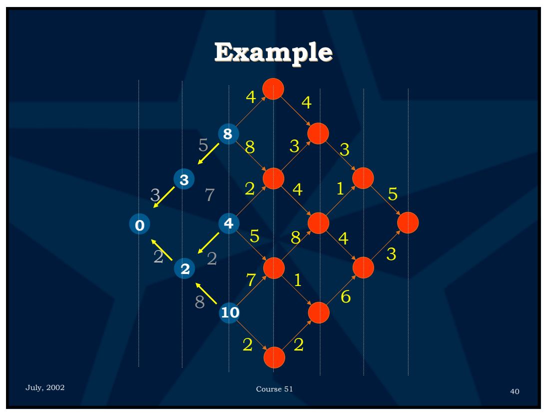


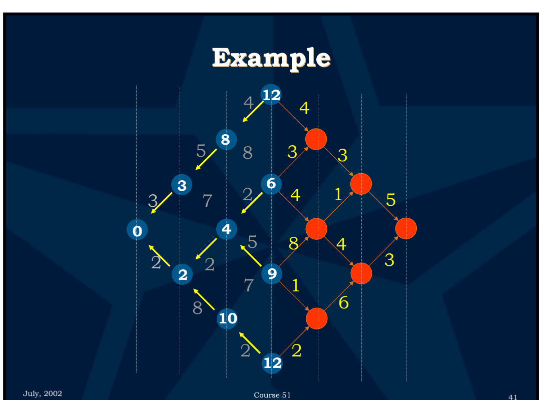
$$f(x) \quad = \min\{f(u_1)+c(u_1,x), f(u_2)+c(u_2,x)\}$$

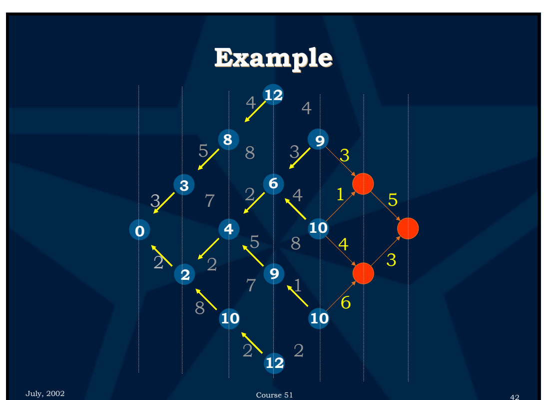
  - total work: $O(n^2)$

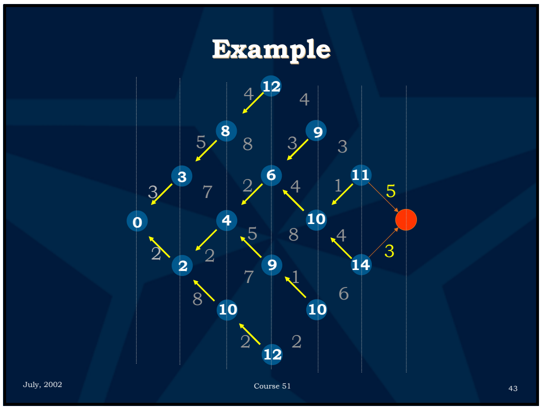


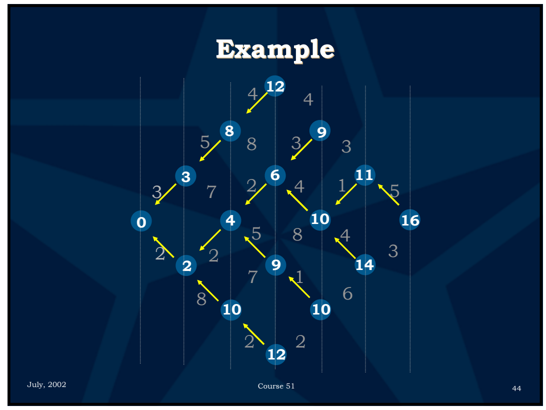


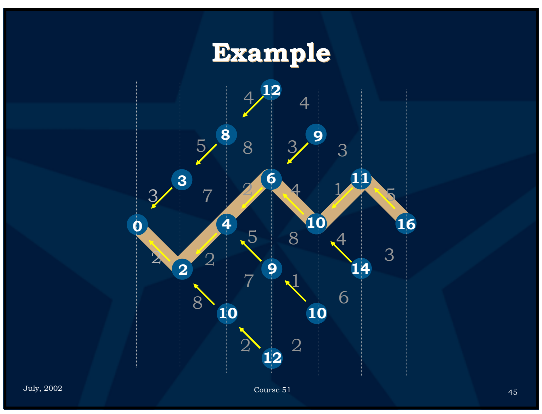












Conclusions

- Many problems in graphics are easily and naturally written as optimization problems.
- Solving them, however, requires skill.
- Continuous vs. discrete

July, 2002 Course 51 46