Combinatorial Optimization
and
Applications in Computer Graphics

Luiz Velho

IMPA – Instituto de Matemática Pura e Aplicada
Rio de Janeiro, Brasil

Outline

- Basic Notions of Combinatorial Optimization
  - Problem Description: Graphs
  - Optimality Principle: Dynamic Programming
  - Shortest Path: Dijkstra Algorithm
- Applications in Computer Graphics
  - Optimal Curves in Maps
  - Surfaces from Countours

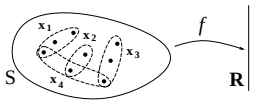
Combinatorial Optimization Problems

- Problem Definition:

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

where $f : S \to \mathbb{R}$, and the set S is:

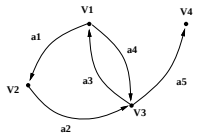
  - *finite*
  - specified by *relations*



Description of S

Graph: $G = (V, A)$

- Vertices: $V = \{v_i \, ; \, i = 1, \ldots N\}$
- Arcs: $A = \{a_{ij} \, ; \, a_{ij} \subseteq V \times V\}$



- Properties
  - Orientation: undirected / directed
  - Topology: acyclic / cyclic

Graph Representation

- Adjacency Matrix: $M_{|V| \times |V|} = (a_{ij}), \quad a_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in A \\ 0 & \text{otherwise} \end{cases}$
- Adjacency List: $L_{|V|} = (\uparrow n(v_i))$
- Implicit Representation: $n(S, v_i)$

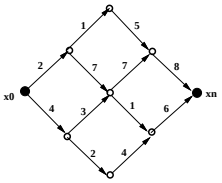
Examples

$$M = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad L = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \begin{matrix} \mapsto \boxed{2} \mapsto \boxed{3} \\ \mapsto \boxed{3} \\ \mapsto \boxed{1} \mapsto \boxed{4} \end{matrix}$$

* *Issues*: space, access time, etc.

Example 1

Find the *path*, $x \in S$, from x_0 to x_n, in $G = (V, A)$
such that the *cost function*, $f(x)$, is minimum.



$x = (x_0, \ldots, x_n)$, with $x_i \in V$ and $(x_i, x_{i+1}) \in A$

$f(x) = \sum c(x_i, x_{i+1})$

Brute Force Method

- Exhaustive Search: *Enumerate all Elements of* S
- Algorithm:

  $c = \infty$
  for all $x \in S$ **do**
    if cost$(x) \leq c$ **then**
      $x_{\min} = x$; $c =$ cost(x);
  return $x_{\min}$

- Size of S is big for most problems of interest

* *Not a practical solution!*

Growth Rates

- Example 1
  - Description Size: $|G| = O(n^2)$
  - Search Space Size: $|S| = \binom{2n}{n} > O(2^n)$

n	n^2	$\binom{2n}{n}$
10	100	184756
20	400	137846528820
30	900	118264581564861424
40	1600	1075072087333606176461620
50	2500	100891344545564193334812497256

* Algorithm *must* be $O(n^k)$
i.e. Polynomial with Description Size

Divide and Conquer Method

- *Split* the problem into simpler problems
- *Solve* simple instances and *Combine* solutions

- Recursive Algorithm:
  if simple instance **then**
  return base case
  else
  return combine(split(p))

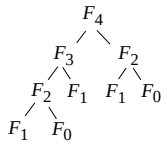
- Iterative Algorithm:
  initial instance ← base case
  while not done **do**
  current instance ← combine(previous instances)
  return final instance

Example 2: Fibonacci Numbers

- Base Cases: $F_0 = 0$, $F_1 = 1$
- Recurrence Relation: $F_n = F_{n-1} + F_{n-2}$

rFib (n)
  if ($n = 0$ or $n = 1$) **then**
  return n
  else
  return rFib($n - 1$) + rFib($n - 2$)

iFib (n)
  $F_0 = 0$, $F_1 = 1$
  for $i = 2, \ldots, n$ **do**
  $F_i = F_{i-1} + F_{i-2}$
  return F_n



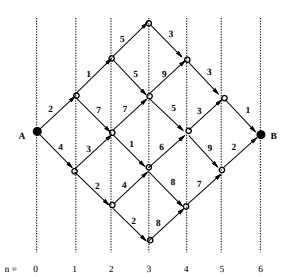
* Complexity:
Recursive: $O(K^n)$, $K \approx 1.6$; Iterative: $O(n)$

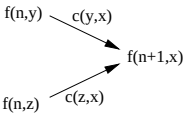
Dynamic Programming

- Recurrence Relation → Iterative Process
  - Sequence of Partial Solutions (steps)
  - Local Decisions (state)
- Define: X_n, state at step n
  $f(n, x)$, partial optimum for $x \in X_n$
  $c(y, x)$, cost of transition $y \mapsto x$
- Algorithm:
  $f(0, x) =$ cost (initial state)
  for $k = 0, \ldots, n$ **do**
  $f(k+1, x) = \min_{y \in X_k}\{f(k, y) + c(y, x)\}$; store $y \mapsto x$
  return backtrack(X_n)
- Complexity: $O(MN)$ N steps, M state size

Example 3: (forward pass)

steps: 0 → 6



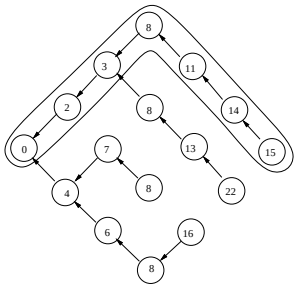


- state variables:
  $x_n := (f(n, x), \uparrow y)$
- recurrence:

$$f(n+1, x) = \min \begin{cases} f(n, y) + c(y, x) \\ f(n, z) + c(z, x) \end{cases}$$

Example 3: (backward pass)

- Backtrack



↙↖ shortest path

○ ← minimum cost

Principle of Optimality

Proposition: *The optimal sequence of decisions has the property that all intermediate decisions must also be optimal.*

- Issues
  - Description:
    sequential order ↔ structure of elements
  - Size:
    local state ↔ consequence of decision chain

Summary

- Methodology
  1. Formulate solution as a *recurrence relation*
  2. Specify *evaluation order* to bulid state
  3. Show that *state size* is bounded by a polynomial
- Limitations
  - Problem must have a natural order
  - Number of state variables should be small

Combinatorial Problems

- Polynomial–Time
  - Linear Partition
  - *Shortest Path*
  - Maximum Flow
  - Bipartite Matching
  - Knapsack
- Hard Problems
  - Traveling Salesman
  - Hamiltonian Path

Shortest Path Problems

Given $G = (V, A)$, find the shortest path from s to t, with $s, t \in V$.

- Conditions (Order)
  - Directed Graph
  - Possibly with Cycles
  - No Cycles with Negative Cost
- Algorithms
  - (DAG): Breadth First Search
  - (DG), $c(i,j) \in \mathbb{R}$: Bellman
  - (DG), $c(i,j) > 0$: Dijkstra

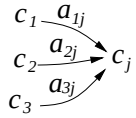
* All pairs shortest path

Bellman Algorithm

- Bellman Equation:

$$c_j = \min_{i \neq j}\{c_i + a_{ij}\}$$

lenght of shortest path from 1 to j

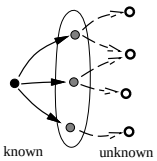


- Algorithm

  $c_1(1) = 0$
  $c_1(j) = a_{1j}, \forall j \neq 1$
  for $k = 2$ to $|V|$ **do**
  $c_k(j) = \min_{i \neq j}\{c_{k-1}(i) + a_{ij}\}$

- Complexity: $O(|V||A|)$

* *Front Propagation*



Dijkstra Algorithm

- Non-negative costs $\Rightarrow c_k(x)$ is monotonic

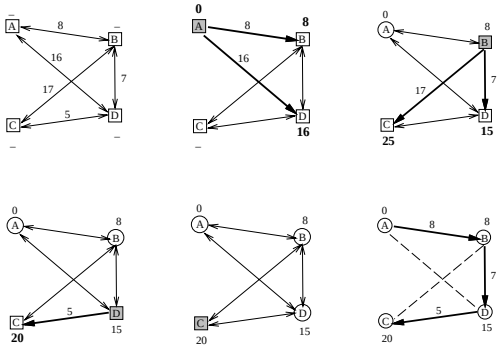
* Search for Path with Minimum Cost

- Algorithm

  $c_1(1) = 0; \quad c_1(j) = a_{1j}, \forall j \neq 1$
  known $= \{1\}$
  while $\overline{\text{known}} \neq \emptyset$ **do**
  pick k, s.t. $c(k) = \min_{j \in \overline{\text{known}}} c(j)$
  known $=$ known $\cup \{k\}$
  for all $j \in \overline{\text{known}}$ **do**
  $c(j) = \min\{c(j), c(k) + a_{kj}\}$

- Complexity: $O(|A| \log |V|)$

Example 4



Applications in Computer Graphics

- Geometric Modeling

* *Optimal Shapes*

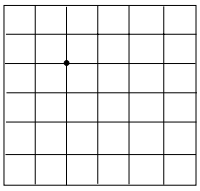
- Curves
  - Shortest Paths in Maps

    (Geographic Information Systems)
- Surfaces
  - Surfaces from Contours

    (Medical Applications)

Shortest Paths in Maps

- Satellite Images $\rightarrow$ Discretization: $N \times M$ grid

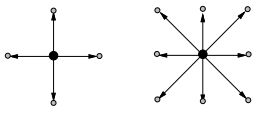


- Minimun cost path from $p = (i,j)$ to $q = (k,l)$
- Cost function: GIS Classification

Neighborhood Graphs

- Discrete Neighborhoods: 4–Connected, 8–Connected

$$\begin{aligned} a_l &= ((i,j),(i+1,j)) \\ a_r &= ((i,j),(i-1,j)) \\ a_t &= ((i,j),(i,j+1)) \\ a_b &= ((i,j),(i,j-1)) \end{aligned}$$

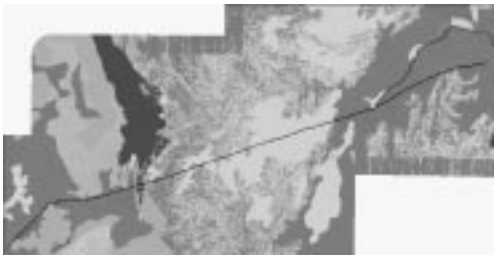


- Cost of Arc: $a = ((i,j),(k,l))$

$$c_a((i,j),(k,l)) = \frac{c(i,j)+c(k,l)}{2}$$

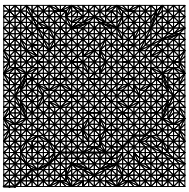
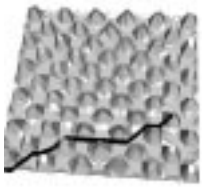
Shortest paths for GIS

- Large Maps: (Tecgraf / Visgraf Project)
  P. Cezar, M. Gattass, L.H. Figueiredo, A. Montenegro, A. Scuri



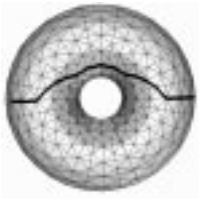
Height Field

- Sinusoidal Peaks

Surface

- Torus

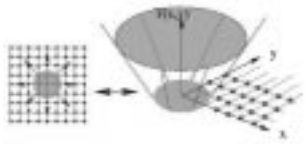



Continuous Front Propagation

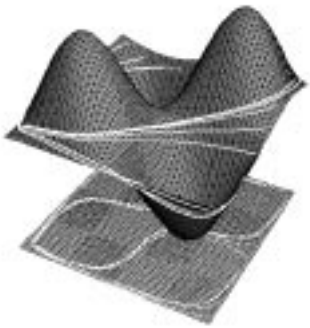
- Eikonal equation: Initial Value Problem

$$\|\nabla T(x,y)\| = 1, \quad \text{with } T(x,y) = 0, (x,y) \in \gamma$$

- Fast Marching Method: Numerical Tracking Technique
  update one grid point at a time → dijkstra algorithm



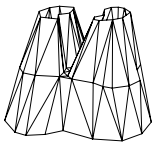
Geodesics on Surfaces (*J. Sethian*)



Surfaces from Contours

- Overview:


(a) cross sections


(b) polygon mesh

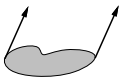

(c) surface

- Surface Fitting Subproblems
  - Correspondence
  - Branching
  - *Tiling*

Applications

- Medical Imaging
  - Reconstruction from CT data

- Geometric Modeling
  - Lofting, Extrusion



Reconstruction from a Pair of Contours

- Contours:

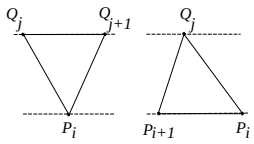
$$P = (p_0, p_1, \cdots, p_{m-1}),\ p_i \in \mathbb{R}^3$$

$$Q = (q_0, q_1, \cdots, q_{n-1}),\ q_i \in \mathbb{R}^3$$

- Elementary Tiles (i.e. *valid triangles*)

$$t_\nabla = (q_j, q_{j+1}, p_i)$$

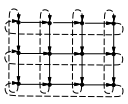
$$t_\Delta = (p_{i+1}, p_i, q_j)$$



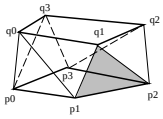
Surface Description through Graphs

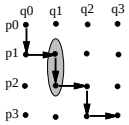
- Toroidal Graph: $G = (V, A)$
  $V = \{v;\ v(i,j) \mapsto (p_i, q_j)\}$

$$A = \{a;\ a(i,j,k,l) \mapsto \left\{ \begin{array}{l} (p_{i+1}, p_i, q_j) \\ (q_j, q_{j+1}, p_k) \end{array} \right\}$$



- Acceptable Surface → Cycle in Graph





The Tiling Problem

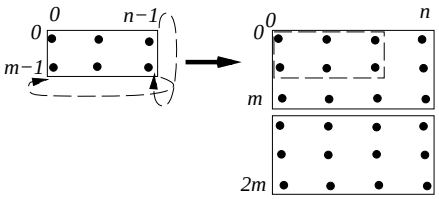
Find the cycle of minimum cost over the toroidal graph

- Cost Function: Measures a *Good* Surface
- Heuristics
  - Maximize Sectional Volume
  - Minimize Total Edge Lenght
  - Minimize Surface Area

$c(i, j, i, j+1) =$ area of triangle (p_i, q_j, q_{j+1})

Reformulating the Optimization Problem

- Graph Transformation: $G = (V, A) \rightarrow G' = (V', A')$



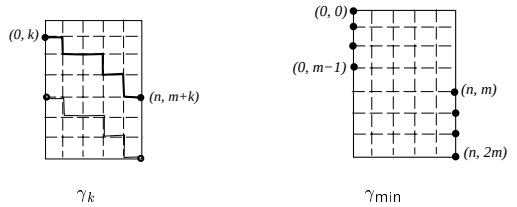
$V' = \{v_{i,j} : i = 0, \ldots, 2m; j = 0, \ldots, n\}$

$c'(v_{i,j}, v_{k,l}) = c(v_{i \bmod m, j \bmod m}, v_{k \bmod m, l \bmod m})$

Shortest Path for Best Surface

- Solution:

$$\gamma_{\min} = \min_{k=0,m-1} \text{OptimalPath}(k, m+k)$$

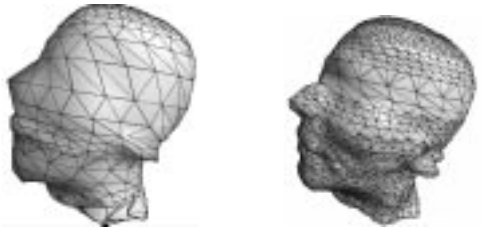


γ_k $\gamma_{\min}$

- Algorithm: version of *all pairs shortest path*

Example

- Head



Other Applications of Shortest Paths

- Computer Vision
  - Stereo Matching
  - Snakes
- Robotics, Animation
  - Path Planning