CSE 6140/ CX 4140:

Computational Science and Engineering ALGORITHMS

Instructor: Anne Benoit

Visiting Associate Professor, CSE

Based on slides by Bistra Dilkina, Jennifer Welch, George Bebis, and Kevin Wayne

Branch-and-Bound

- An enhancement of backtracking
- Applicable to optimization problems (assume we are minimizing)
- Keep track of BEST solution found so far (upper bound on optimal)
- For each node (partial solution), computes a lower bound LB on the value of the objective function for all descendants of the node (extensions of the partial solution)
  - Any extension of this partial solution will have quality at least LB
- Uses the bound for:
  - Ruling out certain nodes as "nonpromising" to prune the tree – if a node's bound is not better than the best solution seen so far
  - Guiding the search through state-space as a measure of "promise"

Branch-and-Bound algorithm

```
Branch-and-Bound(P) // Input: minimization problem P
F ← {(∅,P)}    // Frontier set of configurations
B ← (+∞ , (∅,P))    // Best cost and solution
while F not empty do
  Choose (X,Y) in F – the most "promising" configuration
  Expand (X,Y), by making a choice(s)
  Let (X_1,Y_1), (X_2,Y_2), ..., (X_k,Y_k) be new configurations
  for each new configuration (X_i,Y_i) do
    "Check" (X_i,Y_i)
    if "solution found" then
      if cost(X_i) < B cost then // update upper bound
        B ← (cost(X_i),(X_i,Y_i))
    if not "dead end" then
      if lb(X_i) < B cost then // check lower bound
        F ← F U {(X_i,Y_i)}   // else prune by lb
return B
```

Knapsack (maximization problem)

- Set of items $I_1, \ldots, I_n$; I_i has weight w_i and value c_i
- As many units of each item as we want
- Which items to take so that total weight <= W and total value as large as possible?

- Solution: x_i units of item I_i, for $1 <= i <= n$
- Goal: Maximize sum $x_i c_i$
- Constraint: sum $x_i w_i <= W$

- Running example: 4 items, sorted by nonincreasing c_i/w_i
  - Find $\max(4x_1 + 5x_2 + 6x_3 + 2x_4)$
  - Constraint $33x_1 + 49x_2 + 60x_3 + 32x_4 <= 130$

TSP

- TSP. Given a set of n cities and a pairwise distance function d(u, v), is there a tour of length ≤ D?

- Partial solution (a,T,b): a path from a start node a to b, going through nodes T (same as HamCycle)
- *Choose*: what can be the best-first criteria?
  - The partial assignment with smallest lower bound (most promising of having a short TSP tour)
- *Expand*: choose an edge from b to V-T-{a,b} (same as HamCycle)

- How do we compute a Lower Bound given a partial solution (a,T,b)?

Traveling Salesman Problem—Bounding Function 1

- Because a tour must leave every vertex exactly once, a lower bound on the length of a tour is **the sum of the minimum cost of leaving every vertex.**

- Note: This is not to say that there is a tour with this length. Rather, it says that there can be no shorter tour.

- Given a partial solution (a,T,b)
  - The lower bound = (length of path from a to b) + (the sum of min cost of leaving each vertex in V-T-a)
  - We start with partial solution (v1, $\emptyset$, v1)

Traveling Salesman Problem—Bounding Function 2

- Every vertex must be entered and exited exactly once

- For a given edge (u, v), think of half of its weight as the exiting cost of u, and half of its weight as the entering cost of v

- The total length of a tour = the sum of costs of visiting (entering and exiting) every vertex exactly once.

- A lower bound on the length of a tour is the sum of the lower bound on the cost of **entering and leaving** every vertex**.**
  - Simple: for each vertex, lower bound is the sum of the two shortest adjacent edges divided by 2 (incoming and outgoing)

TSP Bound: Reduced Cost Matrix

Step 1 to reduce: Search each row for the smallest value

- The Cost Matrix for a Traveling Salesperson Problem.

to j

from i

i \ j	1	2	3	4	5	6	7
1	∞	3	93	13	33	9	57
2	4	∞	77	42	21	16	34
3	45	17	∞	36	16	28	25
4	39	90	80	∞	56	7	91
5	28	46	88	33	∞	25	57
6	3	88	18	46	92	∞	7
7	44	26	33	27	84	39	∞

The traveling salesperson optimization problem

Step 2 to reduce: Search each column for the smallest value

- Reduced cost matrix:

i \ j	1	2	3	4	5	6	7	
1	∞	0	90	10	30	6	54	(-3)
2	0	∞	73	38	17	12	30	(-4)
3	29	1	∞	20	0	12	9	(-16)
4	32	83	73	∞	49	0	84	(-7)
5	3	21	63	8	∞	0	32	(-25)
6	0	85	15	43	89	∞	4	(-3)
7	18	0	7	1	58	13	∞	(-26)
								reduced:84

The traveling salesperson optimization problem

j / i	1	2	3	4	5	6	7
1	∞	0	83	9	30	6	50
2	0	∞	66	37	17	12	26
3	29	1	∞	19	0	12	5
4	32	83	66	∞	49	0	80
5	3	21	56	7	∞	0	28
6	0	85	8	42	89	∞	0
7	18	0	0	0	58	13	∞
			(-7)	(-1)			(-4)

The total cost of 84+12=96 is subtracted. Thus, we know the lower bound of feasible solutions to this TSP problem is 96.

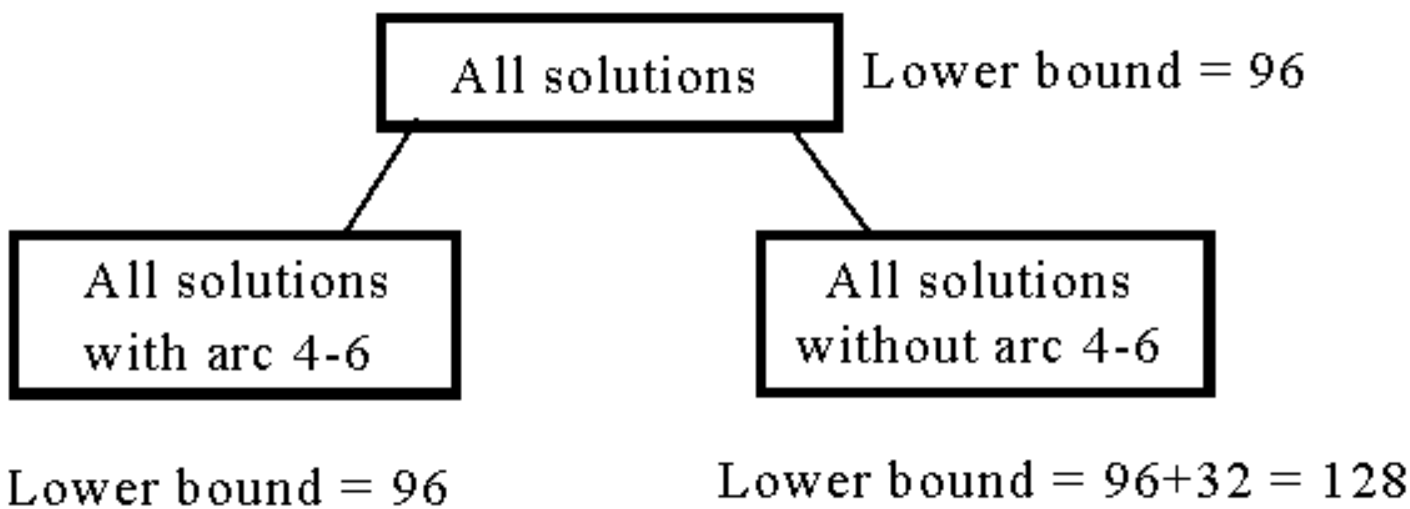
TSP

- How do we branch, i.e., expand the partial solution?

- Any next vertex not chosen yet (multi-way)
- Any next vertex with an edge from the last vertex (multi-way)
- Any next edge connected to last vertex (binary)

The traveling salesperson optimization problem

- Total cost reduced: 84+7+1+4 = 96 (lower bound)
- Decision tree:



The Highest Level of a Decision Tree.

- If we use arc 3-5 to split, the difference on the lower bounds is 17+1 = 18.

For the left subtree
(Arc 4-6 is included)

j / i	1	2	3	4	5	7
1	∞	0	83	9	30	50
2	0	∞	66	37	17	26
3	29	1	∞	19	0	5
5	3	21	56	7	∞	28
6	0	85	8	∞	89	0
7	18	0	0	0	58	∞

A Reduced Cost Matrix if Arc 4-6 is included.

1. 4th row is deleted.
2. 6th column is deleted.
3. We must set c6-4 to be ∞. (The reason will be clear later.)

For the left subtree
(Arc 4-6 is included)

- The cost matrix for all solution with arc 4-6:

i \ j	1	2	3	4	5	7	
1	∞	0	83	9	30	50	
2	0	∞	66	37	17	26	
3	29	1	∞	19	0	5	
5	0	18	53	4	∞	25	(-3)
6	0	85	8	∞	89	0	
7	18	0	0	0	58	∞	

- Total cost reduced: 96+3 = 99 (new lower bound)

For the right subtree
(Arc 4-6 is excluded)

We only have to set c4-6 to be ∞.

i \ j	1	2	3	4	5	6	7
1	∞	0	83	9	30	6	50
2	0	∞	66	37	17	12	26
3	29	1	∞	19	0	12	5
4	32	83	66	∞	49	∞	80
5	3	21	56	7	∞	0	28
6	0	85	8	42	89	∞	0
7	18	0	0	0	58	13	∞

Total cost reduced: 96+32 = 128 (new lower bound)

TSP bounds

- Smarter ideas?
- What if we had a symmetric TSP (the cost of an edge is the same in both directions, e.g., Euclidean distance, then we can treat the graph as undirected)?
- TSP variants:
- Symmetric: distance from u to v = distance from v to u
- Metric: dist(u,v) + dist(v,w) >= dist(u,w)
- Euclidean (cities are represented as (x,y) coordinates and distances are Euclidean in the plane)

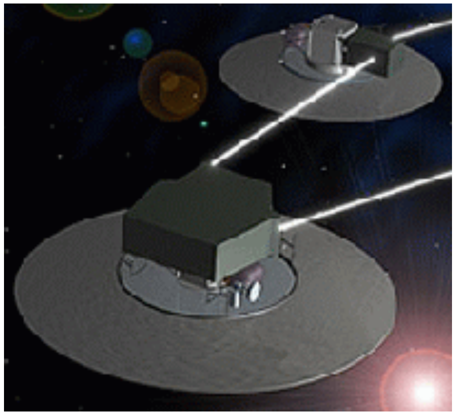
Traveling Salesman Problem (symmetric)— Bounding Function 3 **Dynamic**

- Given a partial solution (a,T,b)
- We have a path from a to b using vertices $T \subseteq V - \{a,b\}$
- A lower bound is the sum of:
  - The partial path we have
  - A lower bound on exiting a and b (their shortest edge **to a vertex in V-T-{a,b}**)
  - A lower bound on visiting the remaining nodes (The cost of the Minimum Spanning Tree for the subgraph over nodes in **V-T-{a,b}**)

DID YOU KNOW THAT

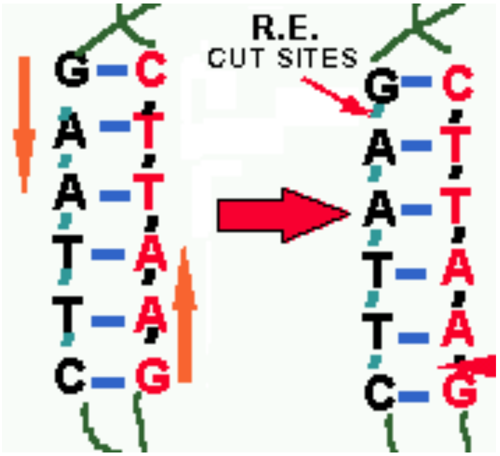
Starlight Interferometer Program

- Use TSP heuristics to:
- Optimize the sequence of celestial objects to be imaged in a proposed NASA *Starlight* space interferometer program.
- Minimize the use of fuel in targeting and imaging maneuvers for the pair of satellites involved in the mission
  - the cities in the TSP are the celestial objects to be imaged,
  - the cost of travel is the amount of fuel needed to reposition the two satellites from one image to the next.
- A team of engineers at Hernandez Engineering in Houston and at Brigham Young University



DNA Universal Strings

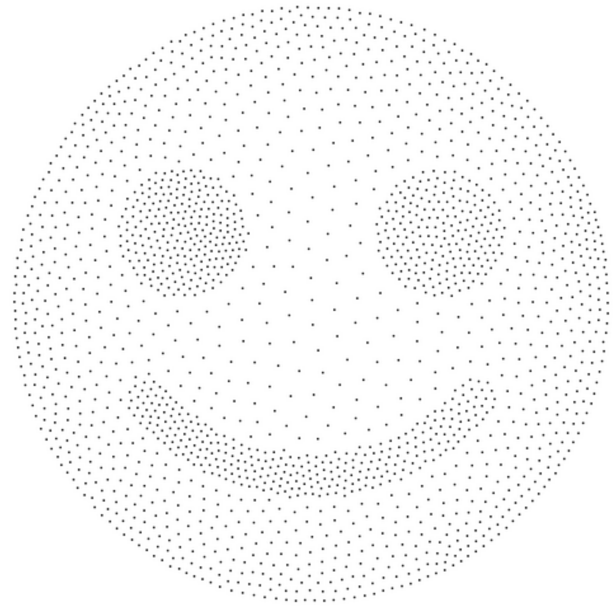
- A group at AT&T to compute DNA sequences in a genetic engineering research project.
- A collection of DNA strings, each of length *k*,
- Need to be embedded in one universal string (that is, each of the target strings is contained as a substring in the universal string),
- With the goal of minimizing the length of the universal string.
  - The cities of the TSP are the target strings, and
  - The cost of travel is *k* minus the maximum overlap of the corresponding strings.

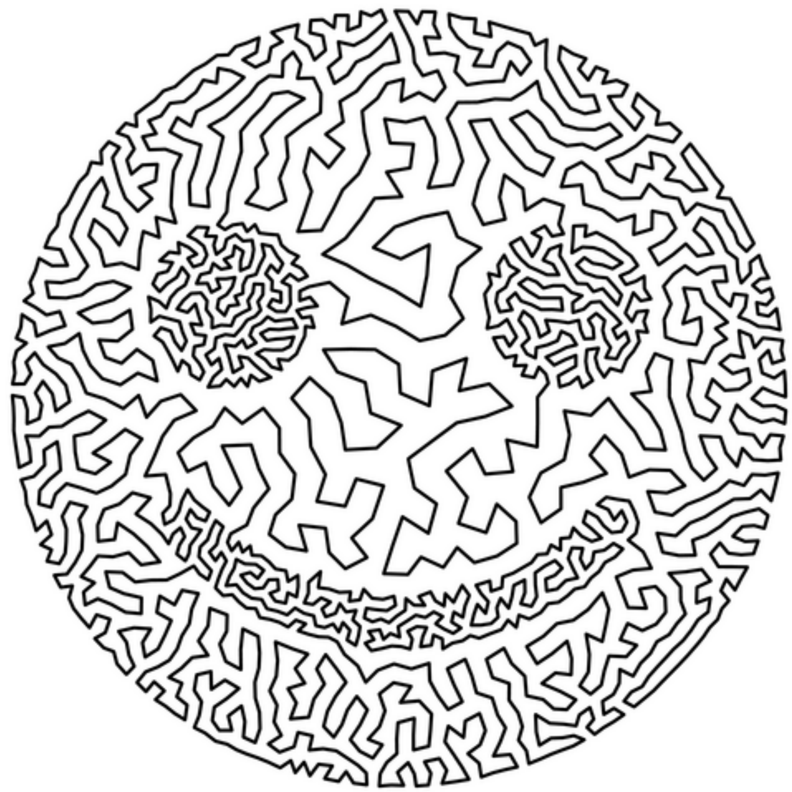


Other Applications

- X-ray crystallography
  - Cities: orientations of a crystal
  - Distances: time for motors to rotate the crystal from one orientation to the other
- High-definition video compression
  - Cities: binary vectors of length 64 identifying the summands for a particular function
  - Distances: Hamming distance (the number of terms that need to be added/subtracted to get the next sum)

TSP Art: Robert Bosch and Craig Kaplan





$1000 prize for finding optimal solution

- Robert Bosch created a 100,000-city TSP instance of Leonardo da Vinci's Mona Lisa