Georgia Tech

CSE 6140/ CX 4140
Computational Science and Engineering
ALGORITHMS

Coping with NP-completeness - 3
Branch and Bound, Local Search

Instructor: Xiuwei Zhang

Assistant Professor

School of Computational Science and Engineering

Based on slides by Prof. Ümit V. Çatalyürek and Bistra Dilkina

Schedule of this lecture

Revisit branch-and-bound: algorithm, concepts

Revisit bounding functions for TSP:

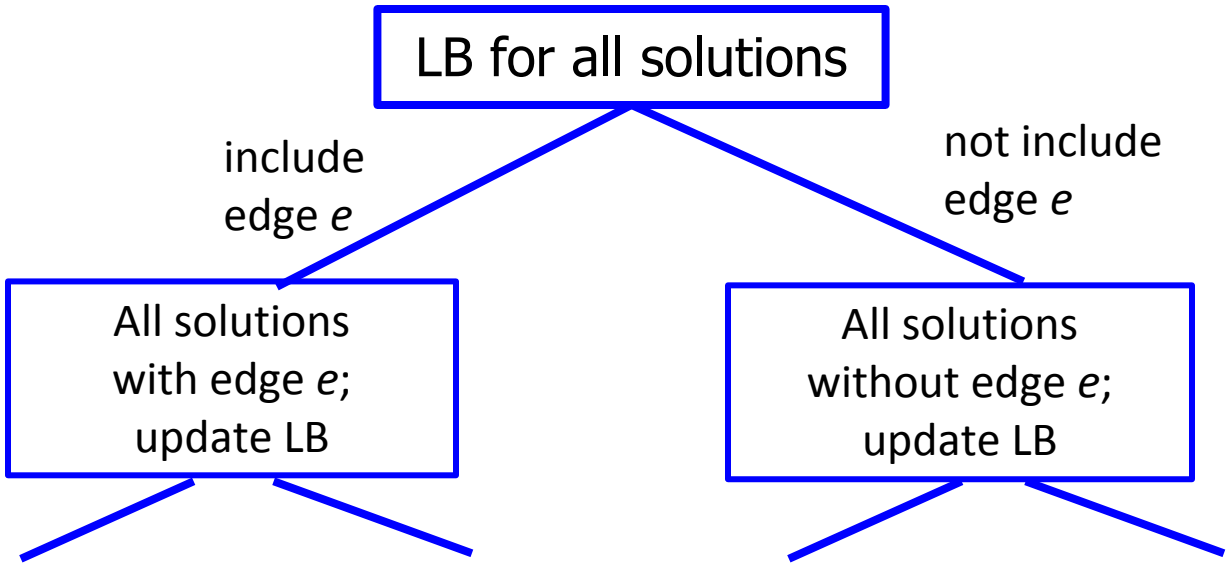
bounding function 2 vs reduced cost matrix method

Start local search

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

Upper bound vs lower bound

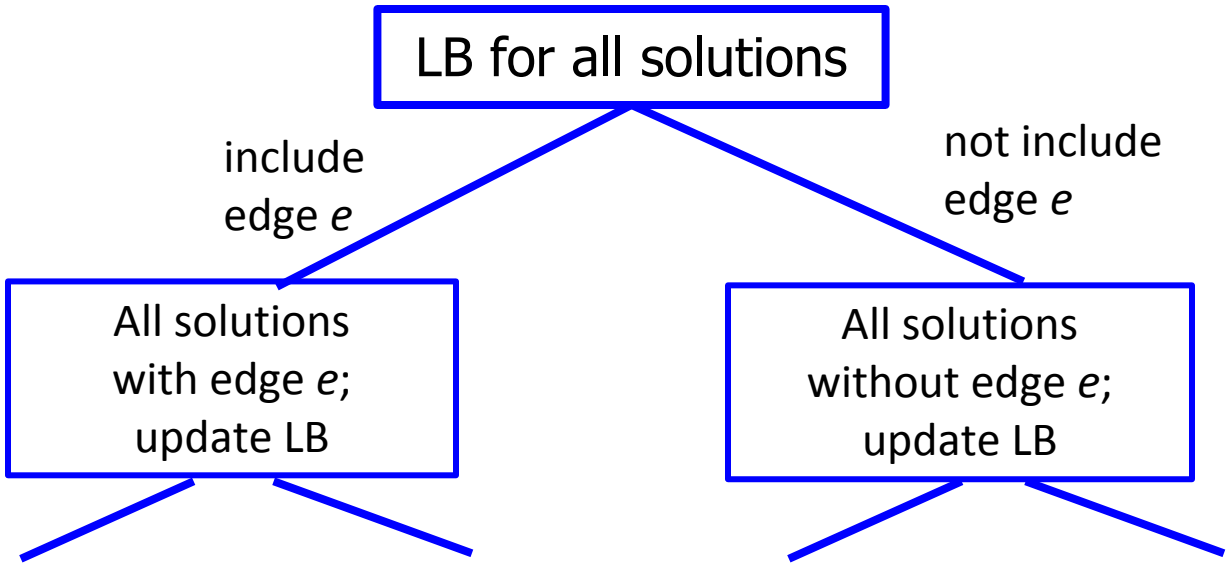


Upper bound
(global variable)

Branch-and-Bound algorithm

Branch-and-Bound(P) // Input: minimization problem P
01 F <- {(∅,P)} // Frontier set of configurations
02 B <- (**+∞ , (∅**,P)) // Best cost and solution
03 **while** F not empty **do**
04 *Choose* (X,Y) in F – the most ”promising” configuration
05 *Expand* (X,Y), by making a choice(es)
06 Let (X_1, Y_1), (X_2, Y_2), ..., (X_k, Y_k) be new configurations
07 **for** each new configuration (X_i,Y_i) **do**
08 ”*Check*” (X_i,Y_i)
09 **if ”**solution found” **then**
10 **if** cost(X_i) < B cost **then** // update upper bound
11 B <- (cost(X_i),(X_i,Y_i))
12 **if** not ”dead end” **then**
13 **if** *lb*(X_i) < B cost **then** //
14 F <- F U {(X_i,Y_i)} // else prune by lb
15 **return** B

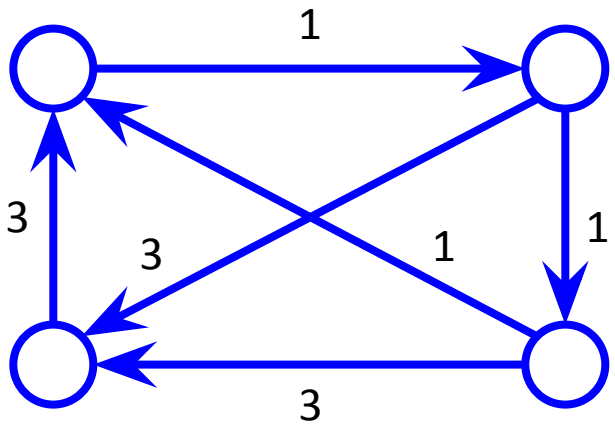
Upper bound vs lower bound



Upper bound
(global variable)

- What is LB used for?

TSP — Calculate Lower Bound



- Most naive lower bound: 0
- Bounding function 1
  sum of the minimum cost of leaving every vertex: 1+1+1+3 = 6
- Bounding function 2
  sum of the two shortest adjacent edges divided by 2 (incoming and outgoing): 2+1+2+3 = 8

TSP bound: reduced cost matrix

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

- The Cost Matrix for a Traveling Salesperson Problem.

from i \ to 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	∞

TSP bound: reduced cost matrix

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

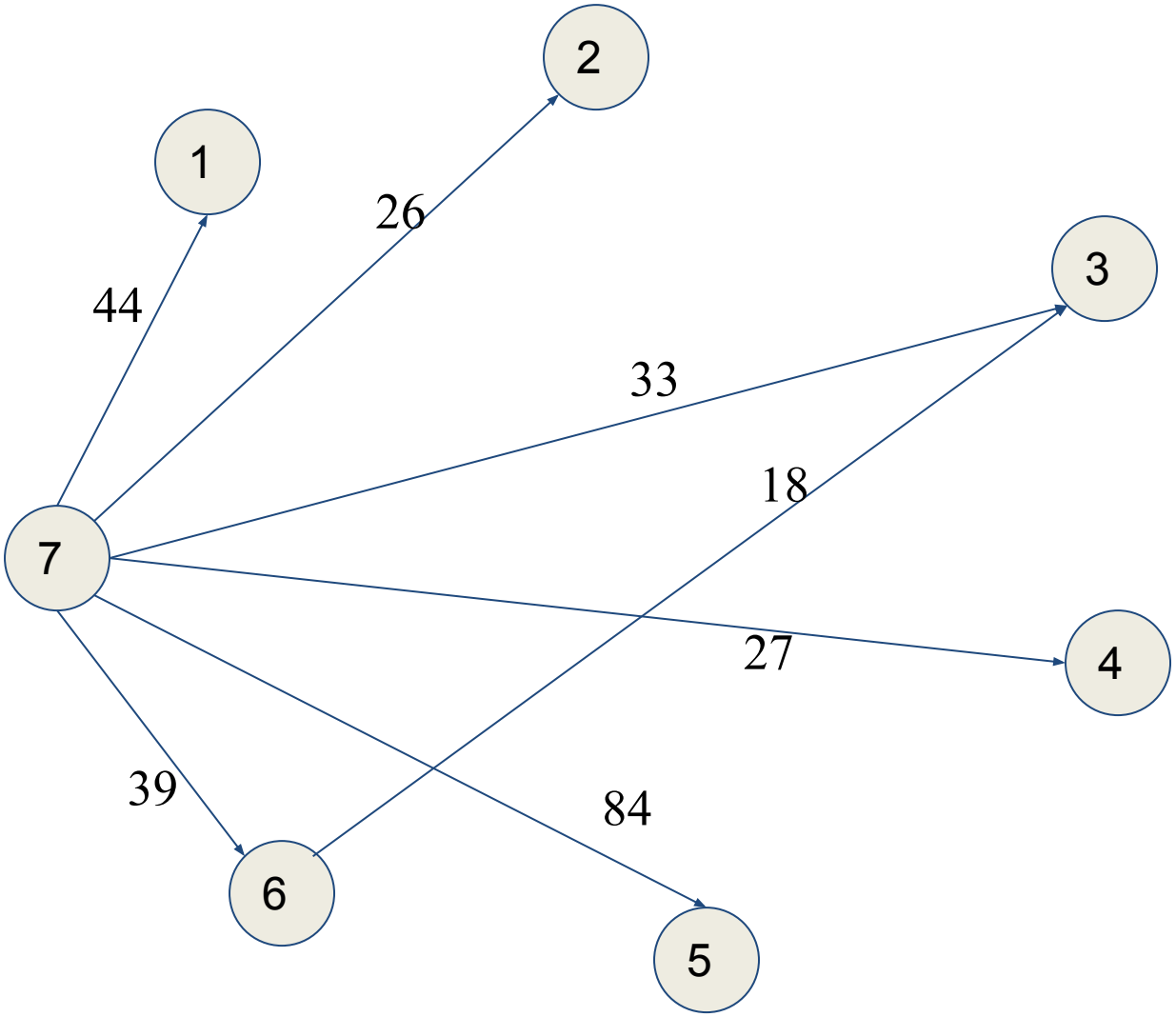
TSP bound: reduced cost matrix

How much **extra** do I pay to enter each vertex?

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	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)

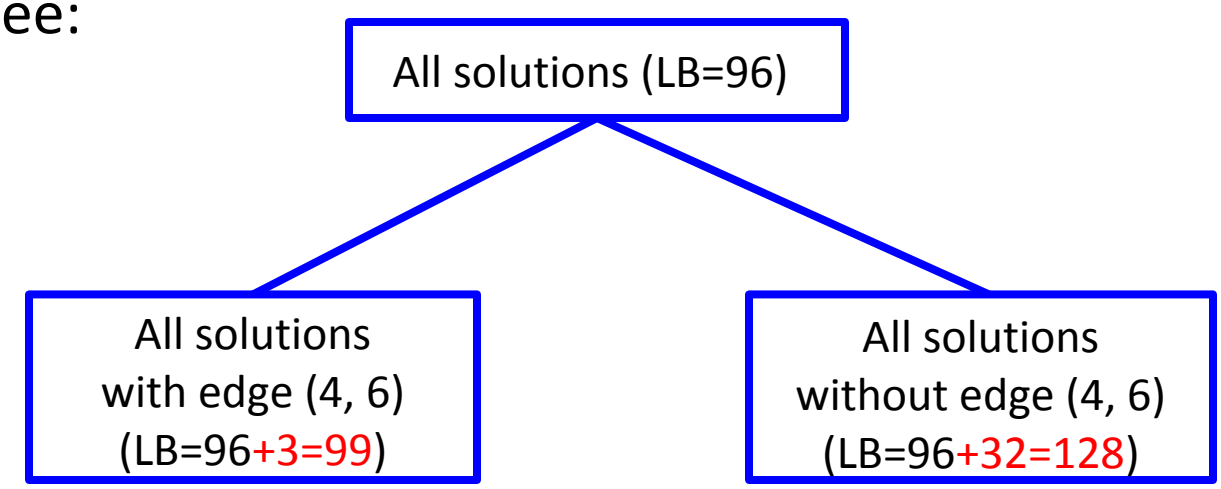
The total cost of 84+12=96 is subtracted. Thus, we know the lower bound of feasible solutions to this TSP problem is 96.

Compare to bounding function 2



TSP bound: reduced cost matrix

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



For the left subtree (Edge (4, 6) is included)

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	3	21	56	7	∞	28
6	0	85	8	∞	89	0
7	18	0	0	0	58	∞

A Reduced Cost Matrix if edge (4, 6) is included.

1. 4^{th} row is deleted.
2. 6^{th} column is deleted.
3. We must set cost(6,4) to be ∞.

For the left subtree (Edge (4, 6) is included)

- The cost matrix for all solution with edge (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 LB)

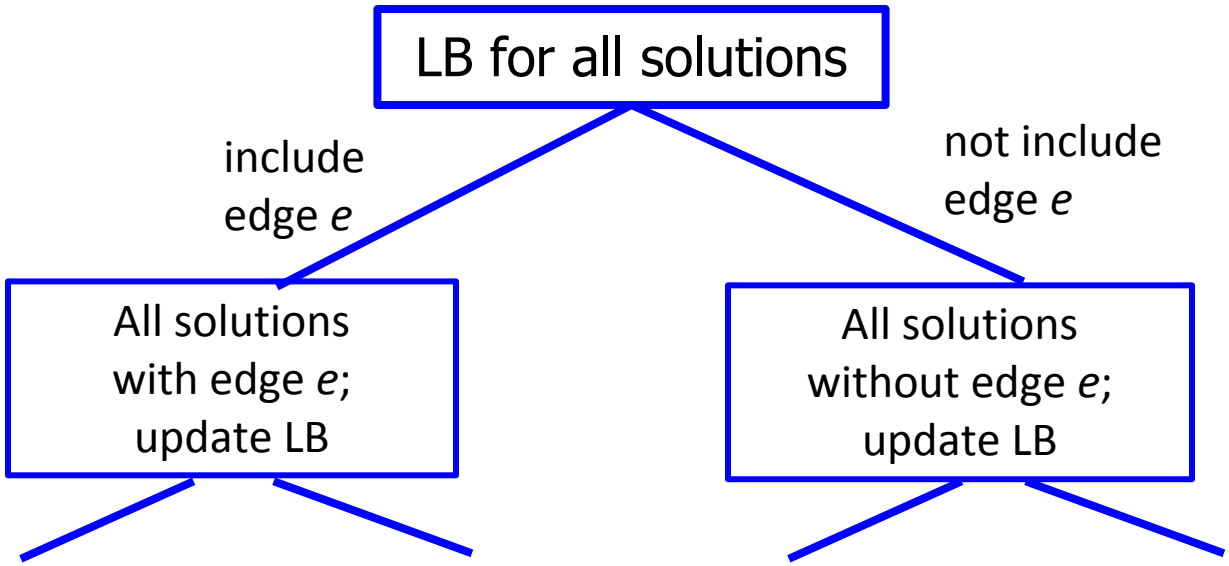
For the right subtree (Edge (4,6) is excluded)

We only have to set cost(4,6) 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 LB)

More on lower bounds



Upper bound (global variable)

- Can LB decrease from root to leaves?
- How does a tight lower bound help?

TSP bounds

- Smarter ideas?

- 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)

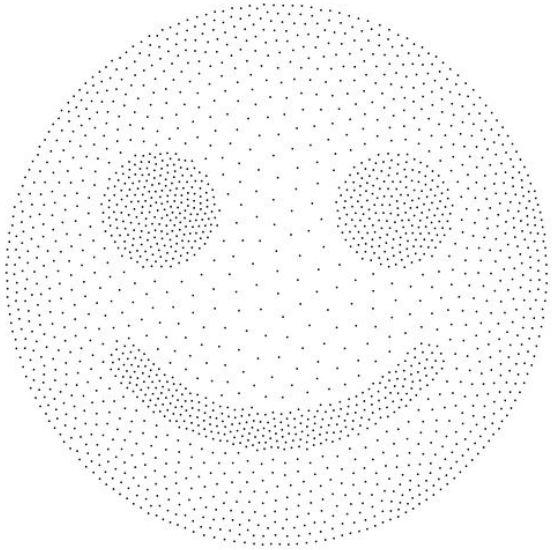
TSP (symmetric)—Bounding Function 3 **Dynamic**

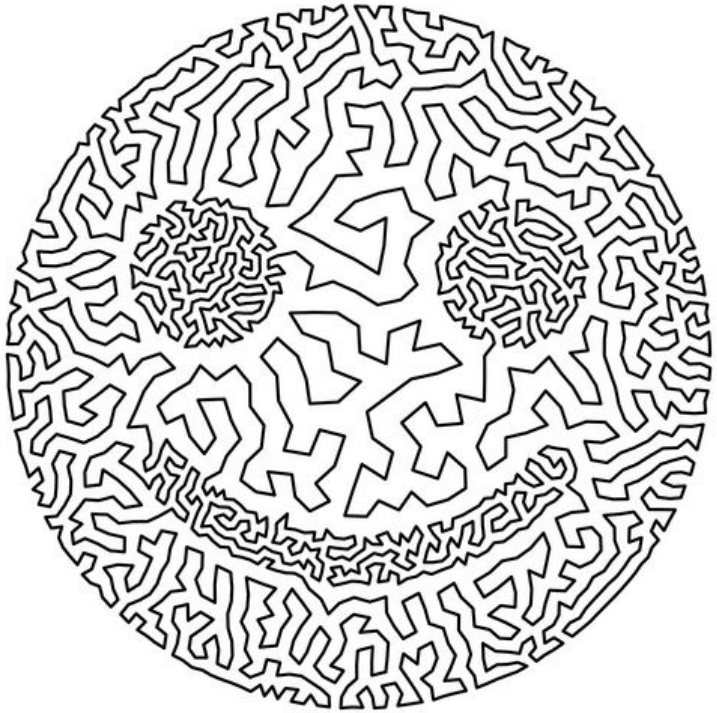
- Given a partial solution (a,T,b)
- We have a path from a to b using vertices $T \subseteq V - \{a, b\}$

- 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}**)

TSP ART

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

HEURISTICS/LOCAL SEARCH

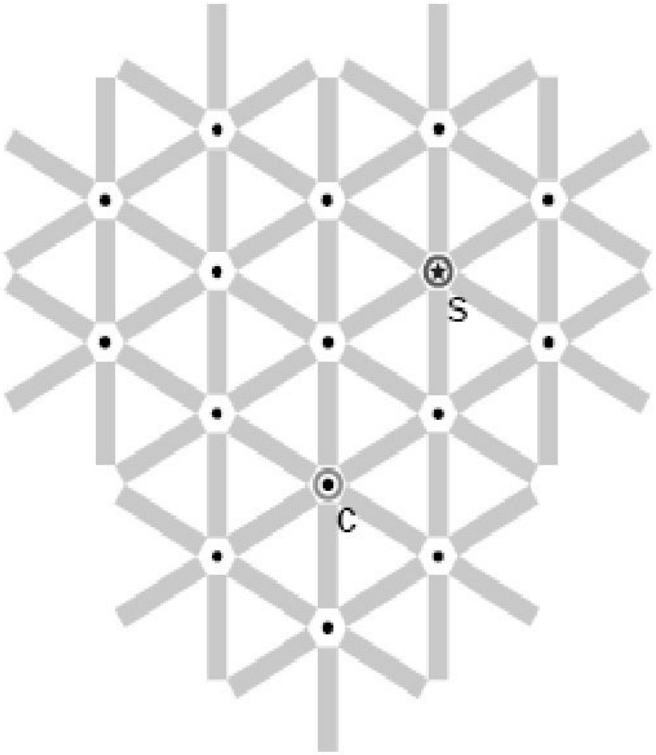
Local Search (LS)

- Start from initial position
- Iteratively move from current position to one of neighboring positions
- Use evaluation function to choose among neighboring positions

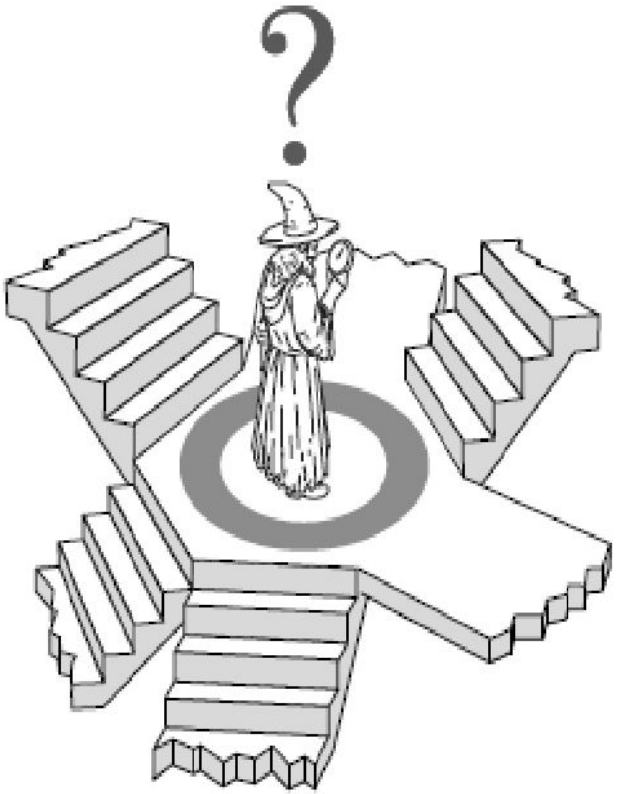
Local Search (LS) Algorithms

- search space S
  - SAT: set of all complete assignments to propositional variables (all "potential solutions")
- solution set $S' \subseteq S$
  - SAT: all satisfying assignments for given formula
- neighborhood relation $N \subseteq SxS$
  - A way to move from one potential solution to another
  - SAT: neighboring variable assignments differ in the value of exactly one variable
- evaluation function g : S $\longrightarrow$ R+
  - SAT: number of clauses satisfied under given assignment

Local Search



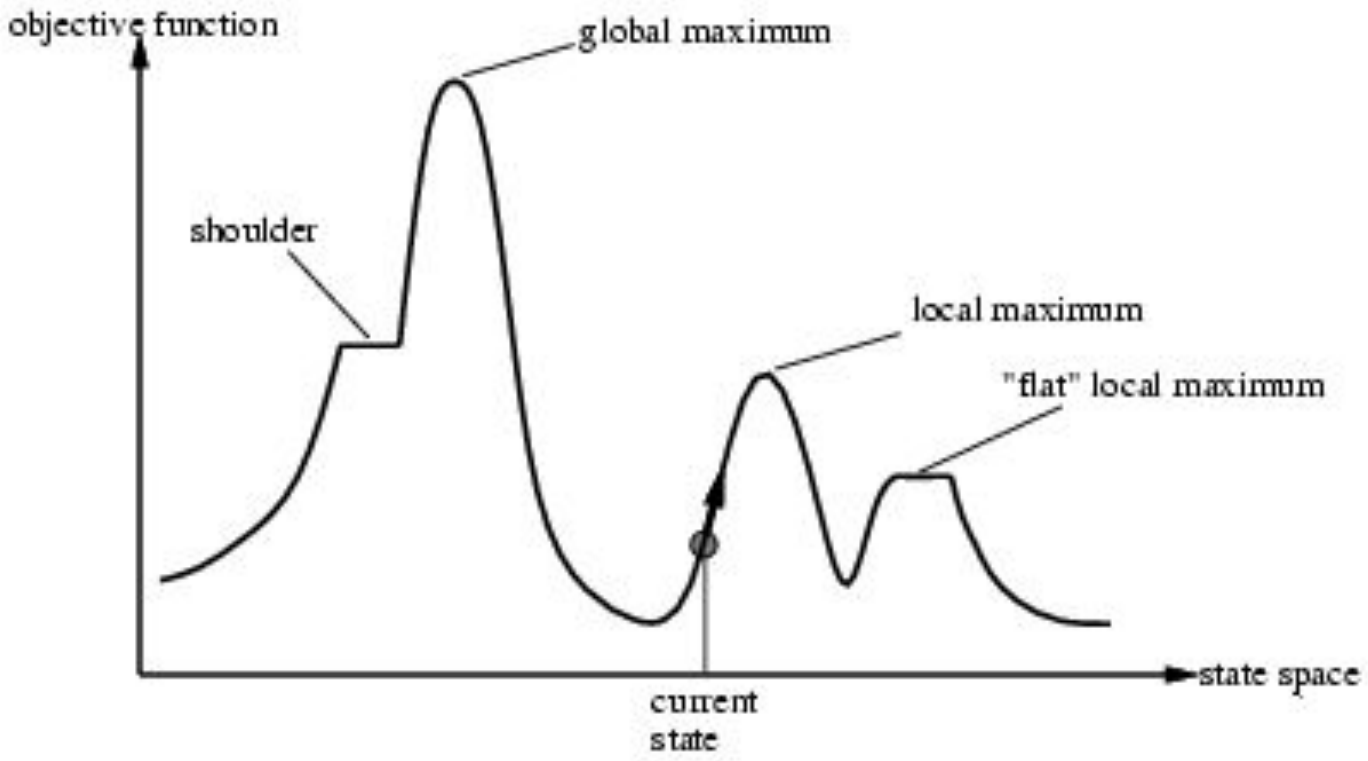
Global view of search space



Local view of search space

State space landscape

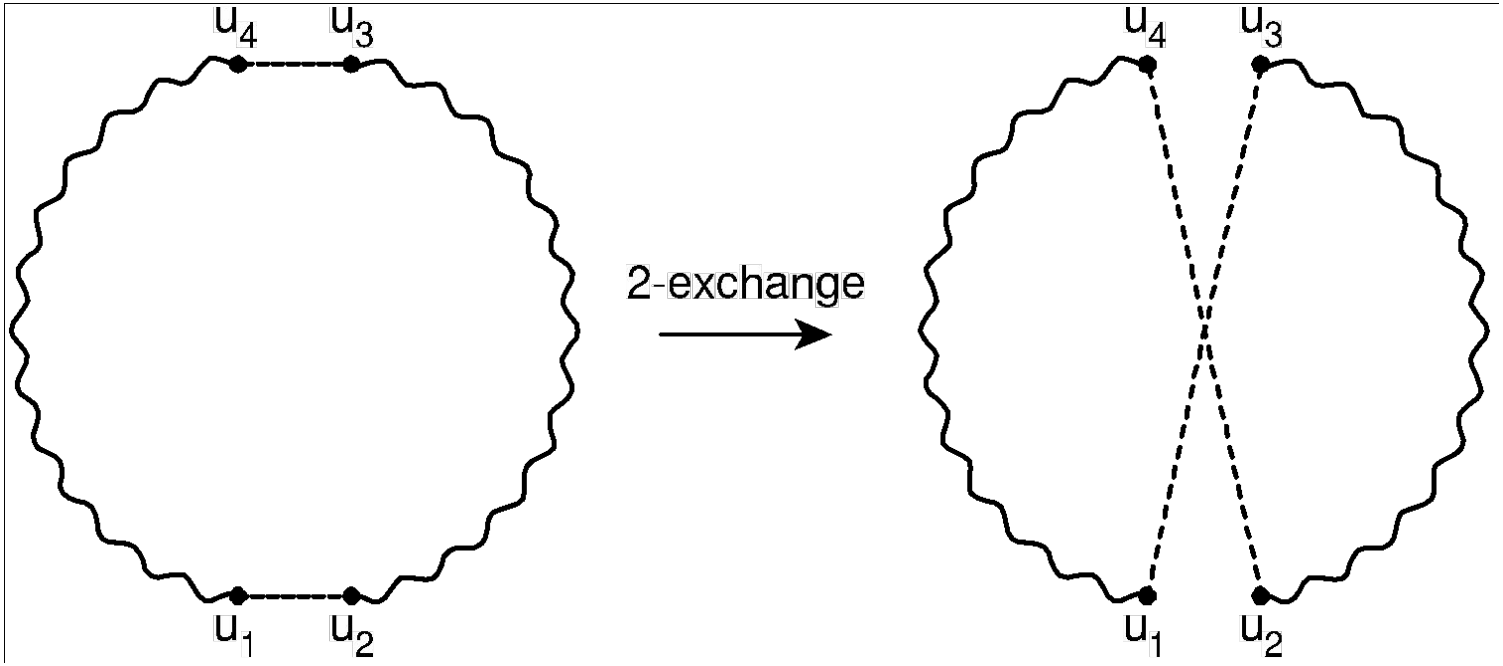
- Objective function defines *state space landscape*



Local Search (LS) Algorithms

- search space S
  - TSP: set of all permutations of vertices (all "potential solutions")
- solution set $S' \subseteq S$
  - TSP: the tours of minimum length
- neighborhood relation $N \subseteq SxS$
  - A way to move from one potential solution to another
  - TSP: neighboring tour differ in several edges
- evaluation function g : S $\longrightarrow$ R+
  - TSP: length of tour

Symm. TSP --- search neighborhood



Search Space: all permutations of the cities (each defines a cycle)
3-opt – delete 3 edges and reconnect fragments into 1 cycle
k-opt – delete k edges and reconnect fragments into 1 cycle