CS 312: Algorithm Analysis

Lecture #27: Traveling Salesperson Problem with Branch and Bound

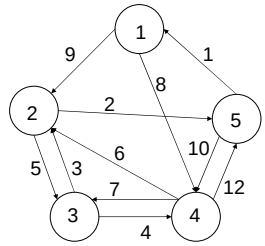
Objectives

Lecture #26: B&B Knapsack, Guided Search	Lecture #27: Traveling Salesperson Problem with Branch and Bound
▪ Implement branch and bound algorithm for Knapsack (last time!) ▪ Understand MiniMax for guided search ▪ Win tic-tac-toe using MiniMax	▪ Introduce the TSP ▪ Learn a more sophisticated bounding function – for the TSP ▪ Reason about Tight Bounds

Traveling Salesperson Problem

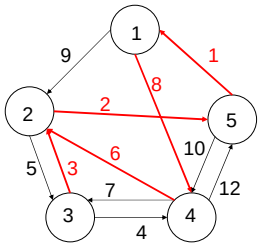
- Hamiltonian Cycle
  - Cycle in the graph that passes through each node exactly once
- Least Cost or "shortest"

Bound on TSP Tour



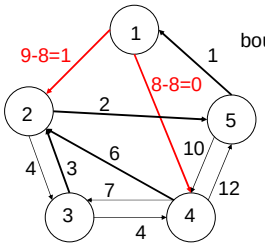
Every tour must leave every node and and arrive at every node.

Bound on TSP Tour



What's the cheapest way to leave each node?

Bound on TSP Tour

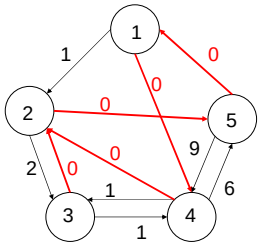


rough draft
bound=8+6+3+2+1

Subtract that cost from each edge leaving a node.

Seeking a tighter lower bound.

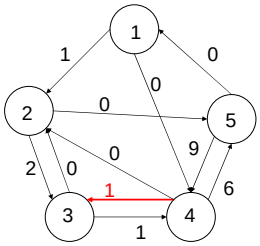
Bound on TSP Tour



bound=20

Save the sum of those costs in the bound (as a rough draft). Subtract each cost from the cost of leaving each individual node.

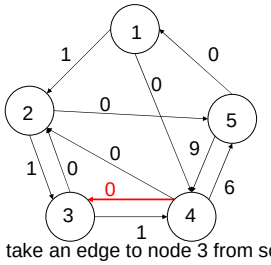
Bound on TSP Tour



bound=20

Does that *tour* (the set of edges now with 0 residual cost) arrive at every node?
In this case, the tour never arrives at node 3.

Bound on TSP Tour



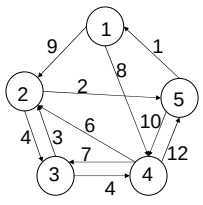
bound=21

We have to take an edge to node 3 from somewhere. Assume we take the cheapest. Subtract its cost from other edges entering node 3 and add the cost to the bound.
We have just tightened the bound.

The Bound

- It will cost at least this much to visit all the nodes in the graph.
  - there's no cheaper way to get in and out of each node.
  - the edges are labeled with the extra cost of choosing another edge.

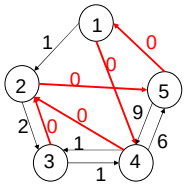
Bound on TSP Tour



999	9	999	8	999
999	999	4	999	2
999	3	999	4	999
999	6	7	999	12
1	999	999	10	999

Algorithms do this using a cost matrix.

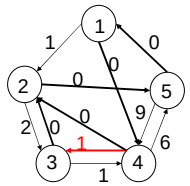
Bound on TSP Tour



999	1	999	0	999	8
999	999	2	999	0	2
999	0	999	1	999	3
999	0	1	999	6	6
0	999	999	9	999	1
					20

Reduce all rows.

Bound on TSP Tour



999	1	999	0	999
999	999	1	999	0
999	0	999	1	999
999	0	0	999	6
0	999	999	9	999

bound: 20 + 1 = 21

Then reduce column #3. Now we have a tight bound.

Using this bound for TSP

start at node 1 in graph (arbitrary)

999	1	999	0	999
999	999	1	999	0
999	0	999	1	999
999	0	0	999	6
0	999	999	9	999

bound = 21

1to2 1to3 1to4 1to5

999	1	999	0	999
999	999	1	999	0
999	0	999	1	999
999	0	0	999	6
0	999	999	9	999

bound = 21+1

infeasible

infeasible

Chose to go from 1 to 2

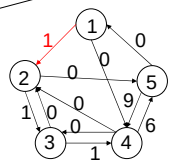
999	1	999	0	999
999	999	1	999	0
999	0	999	1	999
999	0	0	999	6
0	999	999	9	999

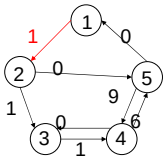
bound = 21

1to2

999	999	999	999	999
999	999	1	999	0
999	999	999	1	999
999	999	0	999	6
0	999	999	9	999

bound = 21+1





Add extra cost from 1 to 2, exclude edges from 1 or into 2.

Chose to go from 1 to 2

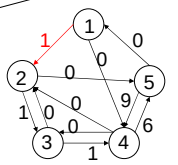
999	1	999	0	999
999	999	1	999	0
999	0	999	1	999
999	0	0	999	6
0	999	999	9	999

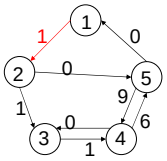
bound = 21

1to2

999	999	999	999	999
999	999	1	999	0
999	999	999	1	999
999	999	0	999	6
0	999	999	9	999

bound = 21+1+1





No edges into node 4 w/ 0 reduced cost.

Chose to go from 1 to 2

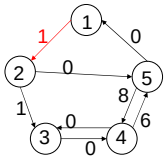
999	1	999	0	999
999	999	1	999	0
999	0	999	1	999
999	0	0	999	6
0	999	999	9	999

bound = 21

1to2

999	999	999	999	999
999	999	1	999	0
999	999	999	0	999
999	999	0	999	6
0	999	999	8	999

bound = 21+1+1



Add cost of reducing edge into node 4.

Bounds for other choices.

1to2,1to4

999	1	999	0	999
999	999	1	999	0
999	0	999	1	999
999	0	0	999	6
0	999	999	9	999

bound = 21

1to2 1to3 1to4 1to5

999	999	999	999	999
999	999	1	999	0
999	999	999	0	999
999	999	0	999	6
0	999	999	8	999

bound = 23

999	999	999	0	999
999	999	1	999	0
999	0	999	1	999
999	0	0	999	6
0	999	999	9	999

bound = 999

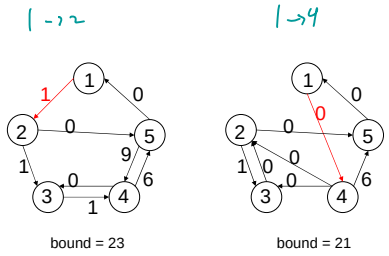
999	999	999	999	999
999	999	1	999	0
999	0	999	999	999
999	0	0	999	6
0	999	999	999	999

bound = 21

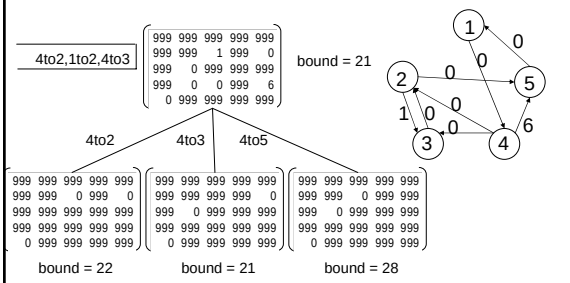
999	1	999	0	999
999	999	1	999	0
999	0	999	1	999
999	0	0	999	6
0	999	999	9	999

bound = 999

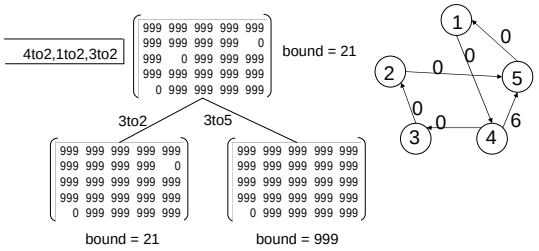
Leaves us with Two Possibilities



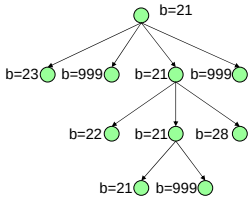
Leaving node 4



Leaving node 3



Call tree for this problem.



Assignment

- HW #20
- See the blog for details