Coping with NP-Hardness

- Strategies to deal with hard problems
- Algorithms for Special Cases
- Fixed Parameter Tractability
- Intelligent Exhaustive Search
  - Backtracking
  - Branch and Bound
- Dynamic Programming based pseudo polynomial algorithm TSP

Imdad ullah Khan

Coping with NP-Hardness

Approaches to tackle hard problems

1. Special Cases: Relevant structure on which the problem is easy
   - Exact results in poly-time only for special cases or a range of parameters
2. Intelligent Exhaustive Search: Exponential time in worst case
   - The base and/or exponent are usually smaller
   - could be efficient on typical more realistic instances
   - Backtracking, Brand-and-Bound
3. Nearly exact solutions: Output is ‘close’ to exact (optimal) solution
   - Approximation Algorithms: Solutions of guaranteed quality in poly-time
   - Heuristic Algorithms: Solutions hopefully good in poly-time
4. Randomized Algorithms: Use coin flips for making decisions
   - Typically used for approximation, also used for easy problems

Branch-and-Bound

A generalization of backtracking for optimization problems

Consider minimization problem ▷ same pattern for maximization

1. Incrementally build solution from partial solutions (of subproblems)
2. Reject a partial solution that will not lead to an optimal solution

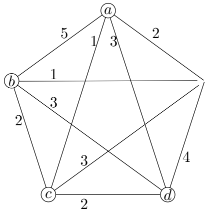
    ▷ As in backtracking, need a basis for eliminating partial solutions

    To reject a partial solution, we must be certain that its cost exceeds that of some known solution (minimization problems)

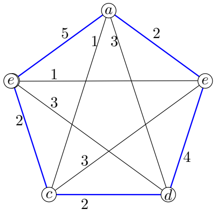
Generally the optimal cost is not known or cannot be computed efficiently, so instead we use a quick lower bound on this cost

Branch and Bound for TSP

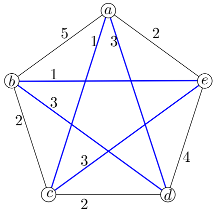
Given a complete graph G on n vertices with edge weights, a TSP tour is a (weighted) Hamiltonian cycle in G



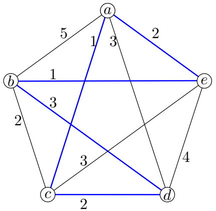
K_5 with edge weights



A TSP tour of length 15



A TSP tour of length 11

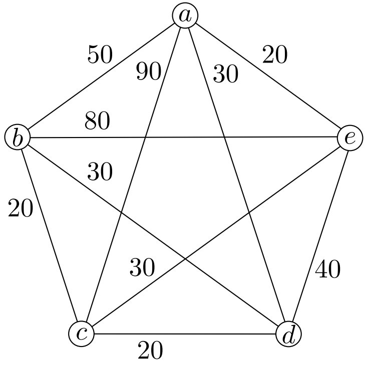


A TSP tour of length 9

Traveling Salesman Problem TSP(G) Given a complete graph G on n vertices with edge weights, find a minimum cost Hamiltonian cycle in G

Branch and Bound for TSP

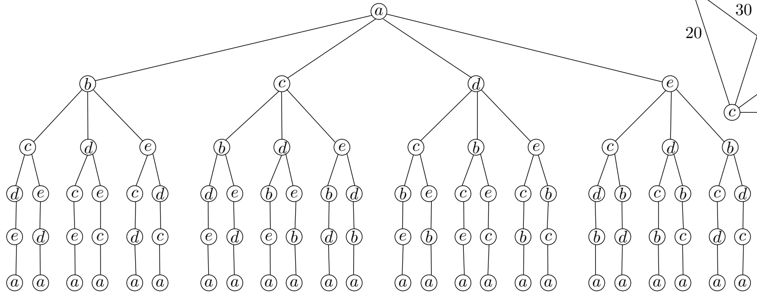
The naive brute-force algorithm checks all possible $(n-1)!$ cycles in G

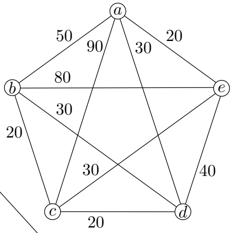


K_5 with edge weights

Branch and Bound for TSP

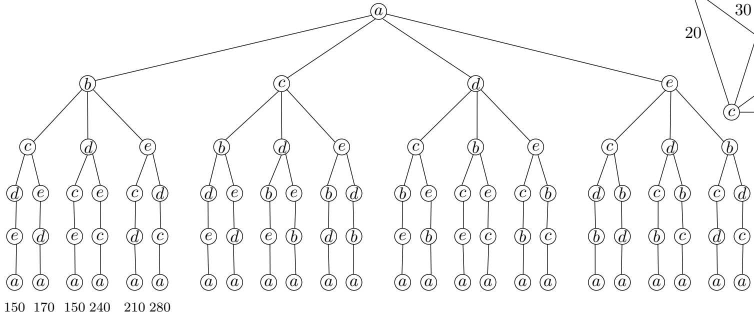
The naive brute-force algorithm checks all possible $(n-1)!$ cycles in G

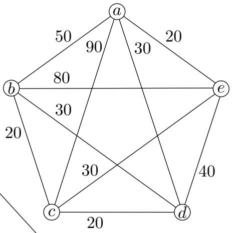




Branch and Bound for TSP

The naive brute-force algorithm checks all possible $(n-1)!$ cycles in G



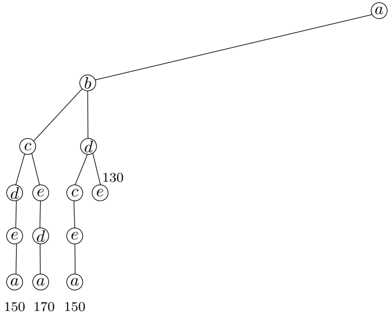


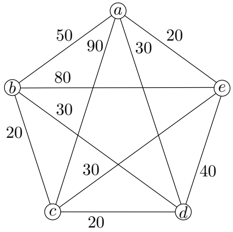
Branch and Bound for TSP

Branch-and-Bound based technique

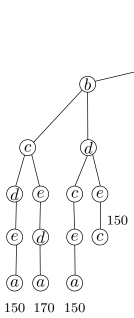
1. Grow a tree of partial solutions (pieces of Hamiltonian paths)
2. At each tree node, check if an extension of this partial solution could be better than the best known solution so far
3. If not, we do not continue the branch

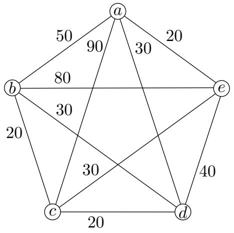
Branch and Bound for TSP



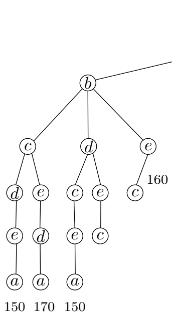


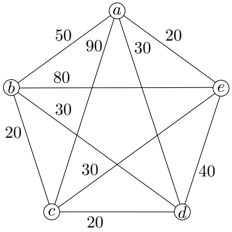
Branch and Bound for TSP



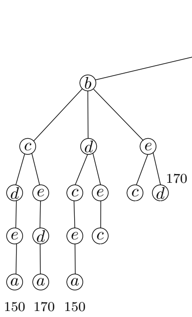


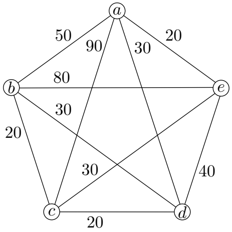
Branch and Bound for TSP



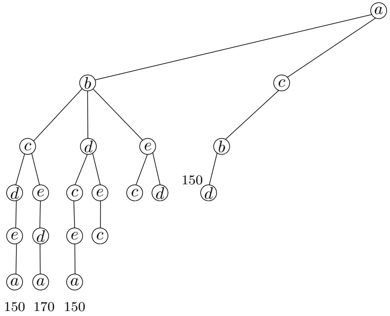


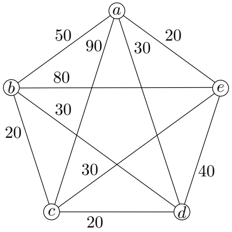
Branch and Bound for TSP



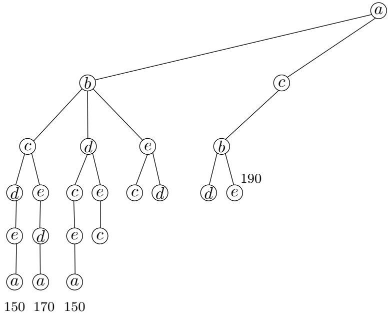


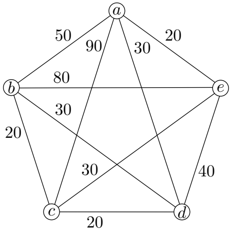
Branch and Bound for TSP



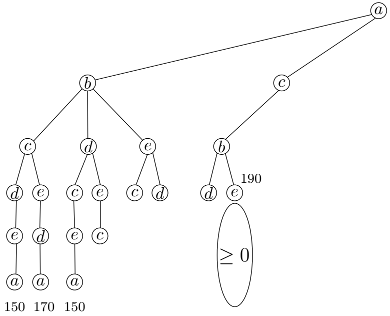


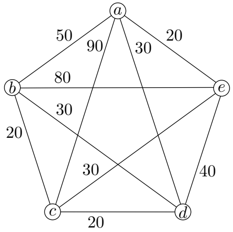
Branch and Bound for TSP



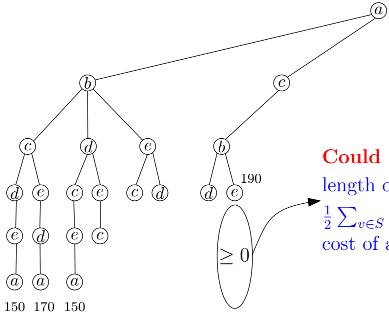


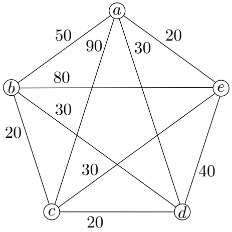
Branch and Bound for TSP





Branch and Bound for TSP





Could use other lower bounds

length of OPT-TSP tour (using a subset of vertices S is at least)

$\frac{1}{2}\sum_{v\in S}$ two min length edges incident on v

cost of a MST of $G[S]$