Artificial Intelligence
人工智慧

Dynamic Programming VS. Ant Colony Optimization (TSP)

Computer Science and Information Engineering
National Taipei University of Technology

Algorithm

- Exact Algorithm
  - Problem-solving method without approximate or error

- Heuristic Algorithm
  - Problem-solving method to produce approximate solutions
    - Select the best solution within an acceptable time and cost
    - Obtain a certain trade-off in complexity and quality of resolution
  - Most are based on an imitation of natural algorithms
    - e.g., Ant Colony Algorithm (ACO), Genetic Algorithm (GA), etc.

Dynamic Programming (DP)

- Dynamic Programming (DP) is an optimization technique that solves problems with overlapping subproblems by breaking them into smaller subproblems, storing results, and avoiding redundant computations to improve efficiency.

- Main idea
  - Set up a recurrence relating a solution to a larger instance to solutions of some smaller instances
  - Solve smaller instances once
  - Record solutions in a table
  - Extract solution to the initial instance from that table

Divide-and-Conquer

- Divide
  - Break down the original problem into smaller subproblems
  - Each subproblem should represent a part of the overall problem
- Conquer
  - If the subproblem is small enough, solve it directly; otherwise, break the subproblem down recursively
- Combine
  - Combine the sub-problems to get the final solution of the whole problem

Traveling Salesman Problem

- Traveling Salesman Problem (TSP)
  - We are given n cities $1, 2, \ldots, n$ and the coordinates or the distance d_{ij} between any two cities i and j
  - The Traveling Salesman Problem (TSP) asks for the total distance of the shortest tour of the cities
    - Each city is visited exactly once
    - At the end, come back to the start city
    - Assume that the distance is equal to the cost, let $d_{ij} = d_{ji}$

DP: Traveling Salesman Problem

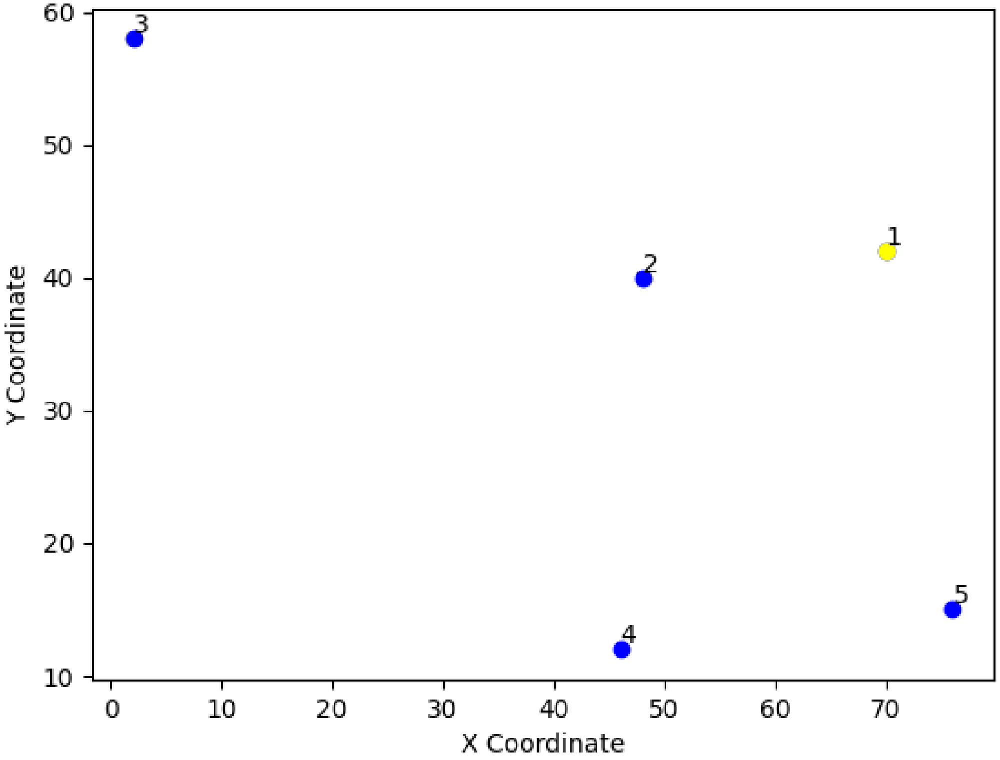
- Let $g(i, S)$ be the length of a shortest path starting at vertex i, going through all unvisited vertices in S and terminating at vertex i.

$$g(i, S) = \begin{cases} \min\limits_{k \in S} \{C_{ik} + g(k, S - \{k\})\} & if\ S \neq \emptyset, \\ C_{is} & otherwise, \end{cases}$$

  - $g(i, S)$ represents the minimum total cost of traveling from city i while visiting all unvisited cities in set S.
  - C_{ik} is the cost of traveling from city i to city k
  - C_{is} is the cost of traveling from city i to start city s

DP: Traveling Salesman Problem

- Cities $n = 5$



DP: Traveling Salesman Problem

- City Coordinates

1	2	3	4	5
(70, 42)	(48, 40)	(2, 58)	(46, 12)	(76, 15)

- Distance d_{ij} between any two cities i and j
  - $d_{ij} = Round(d_{ij}, 0)$

	1	2	3	4	5
1	0	22	69	38	27
2	22	0	49	28	37
3	69	49	0	63	85
4	38	28	63	0	30
5	27	37	85	30	0

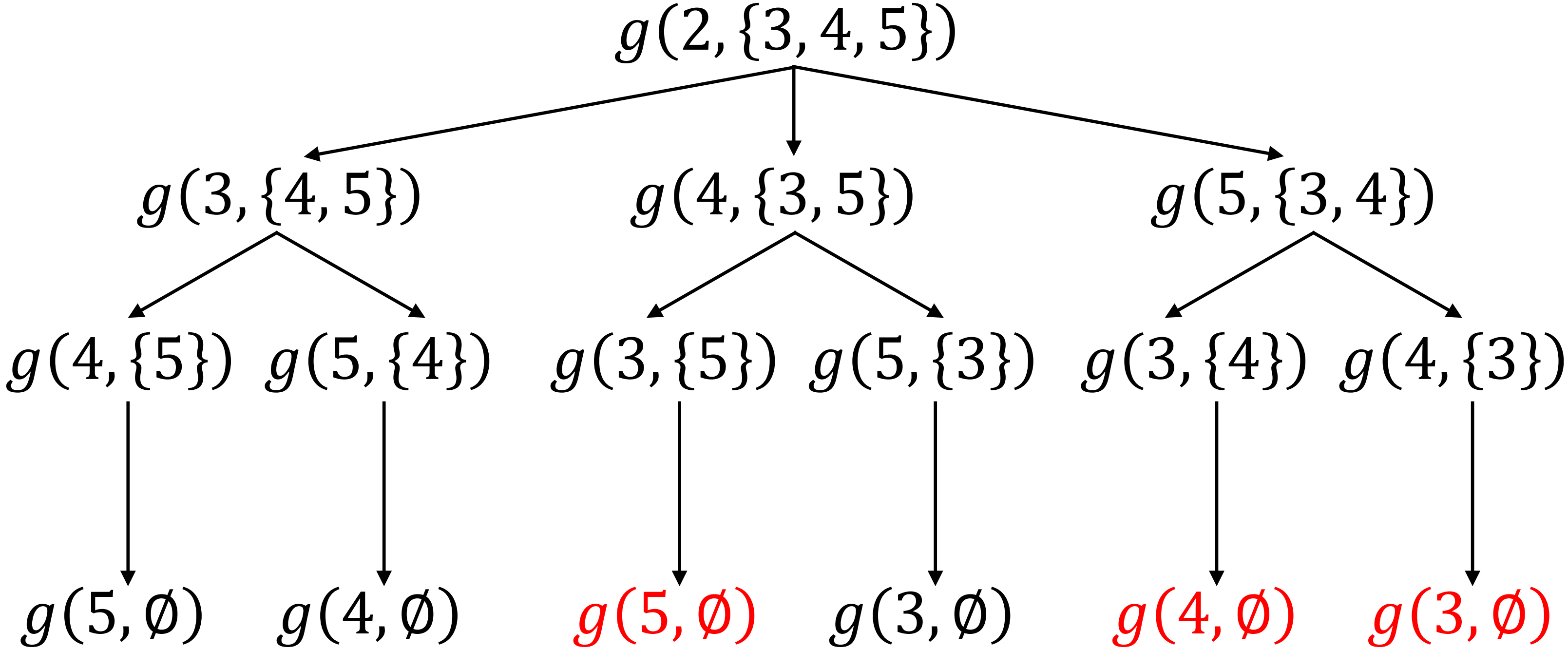
DP: Traveling Salesman Problem

- $i = 1, \;\; S = \{ 2, 3, 4, 5 \}$

$$
\begin{tikzcd}
 & & & g(1, \{2, 3, 4, 5\}) \arrow[dlll] \arrow[dl] \arrow[dr] \arrow[drrr] & & & \\
g(2, \{3, 4, 5\}) & & g(3, \{2, 4, 5\}) & & g(4, \{2, 3, 5\}) & & g(5, \{2, 3, 4\})
\end{tikzcd}
$$

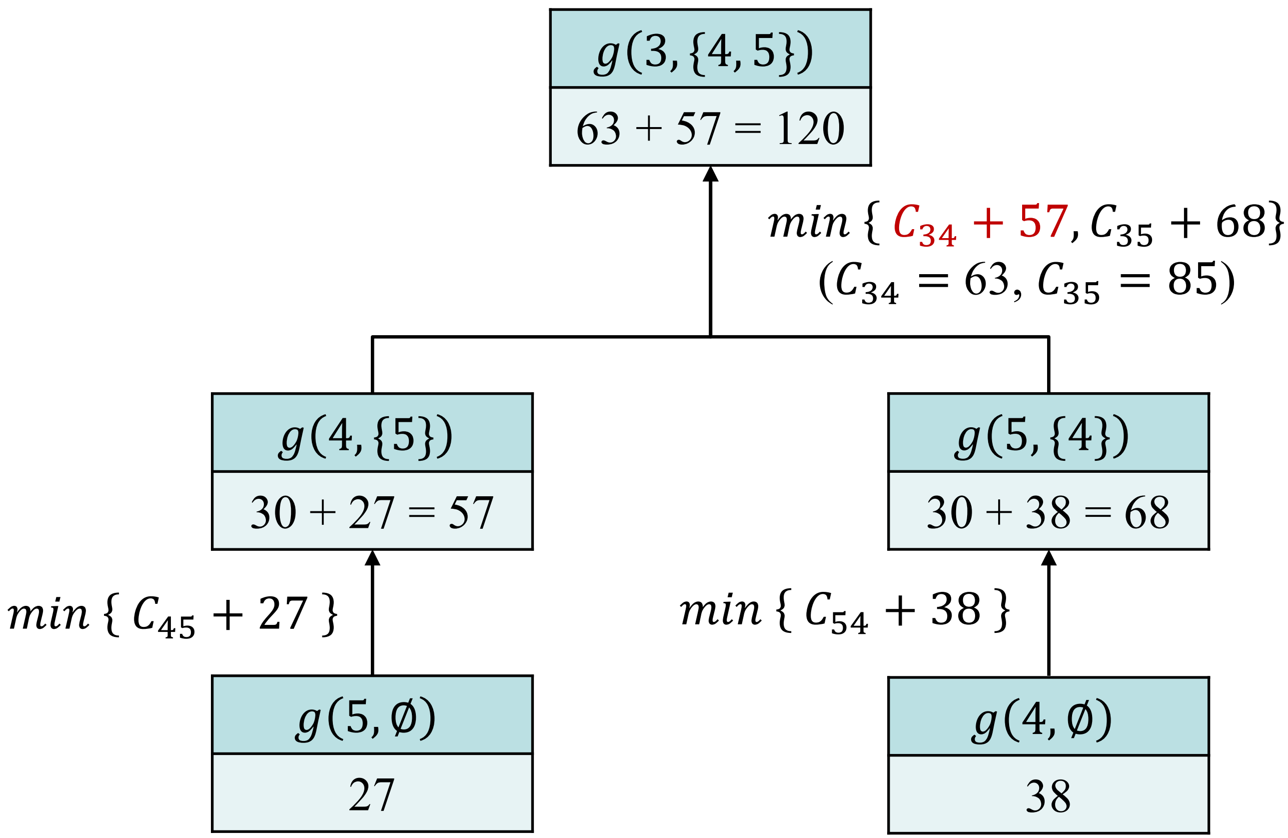
DP: Traveling Salesman Problem

- $i = 2, \quad S = \{3, 4, 5\}$



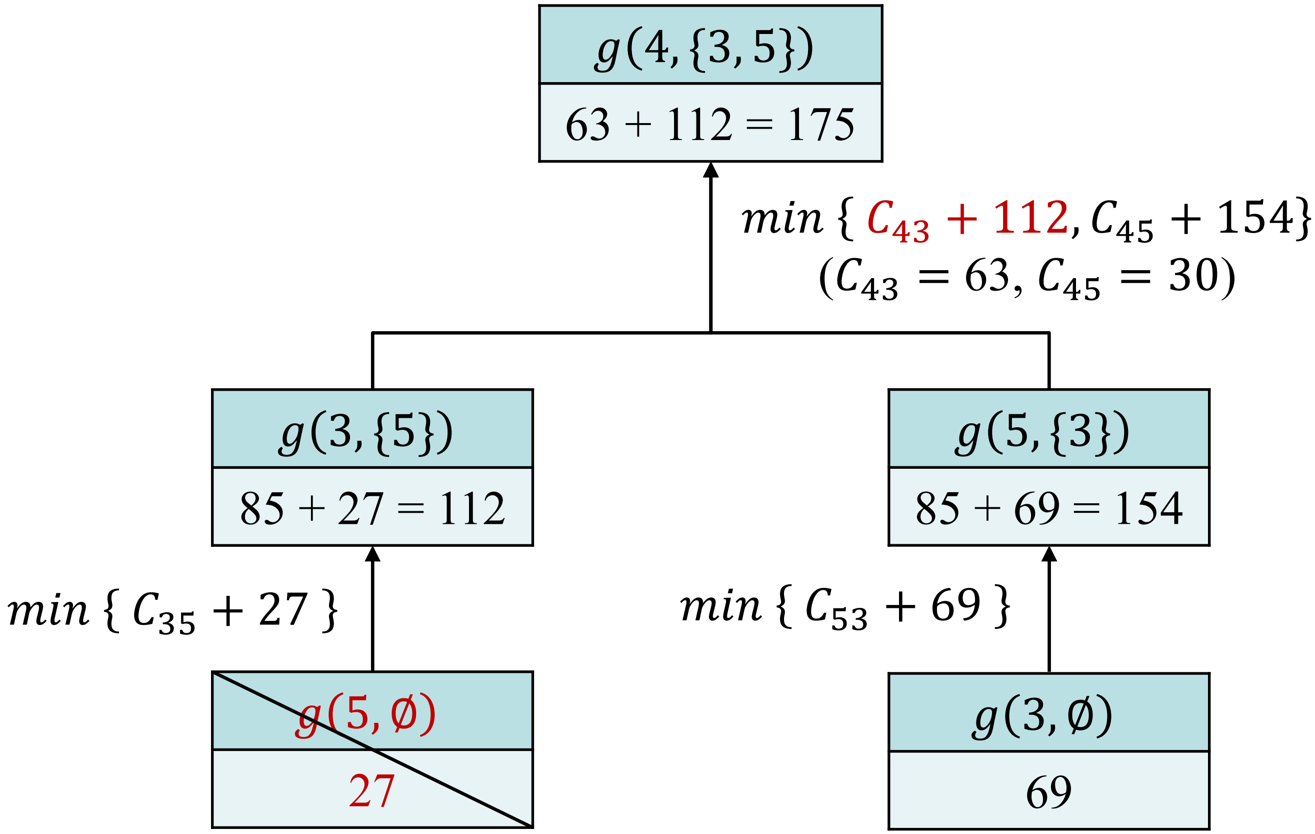
DP: Traveling Salesman Problem

- $i = 3, \ S = \{4, 5\}$



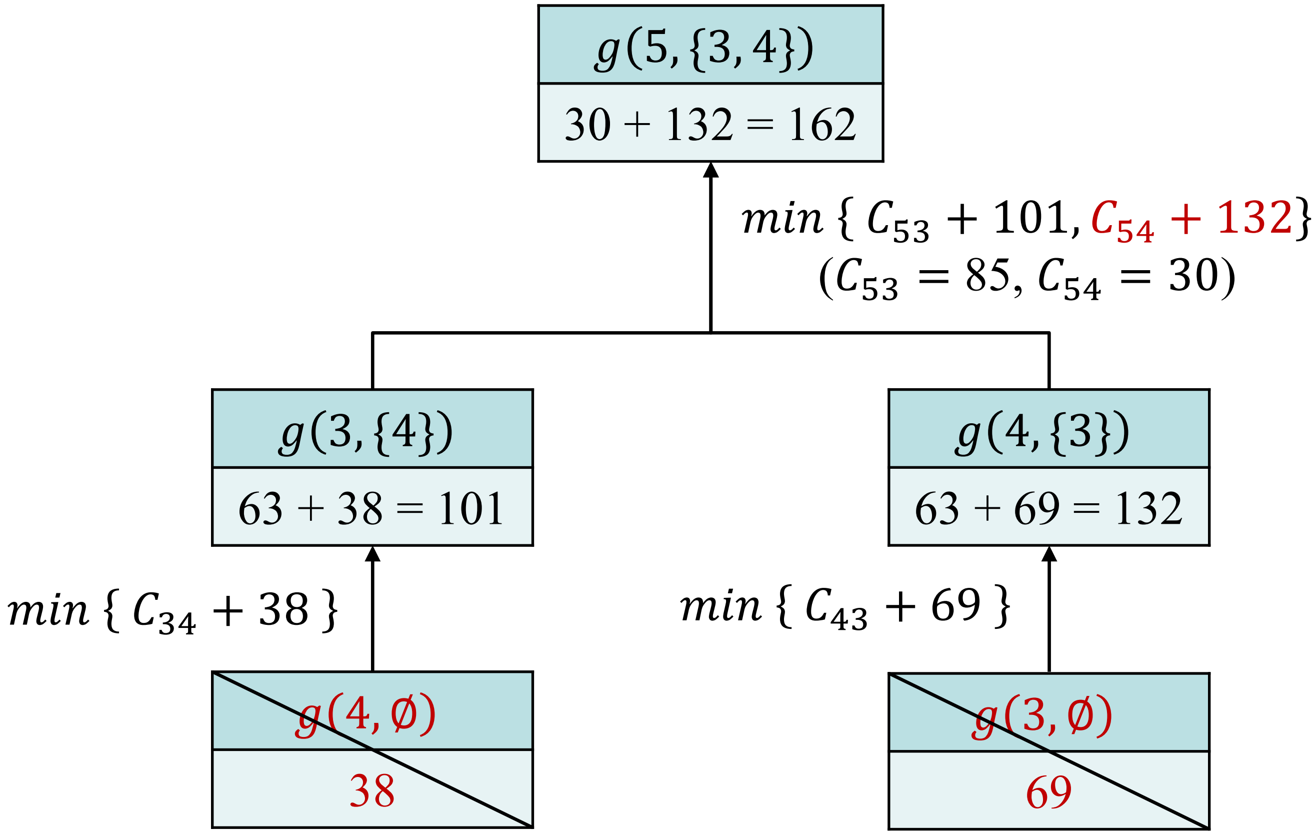
DP: Traveling Salesman Problem

- $i = 4, \ S = \{3, 5\}$



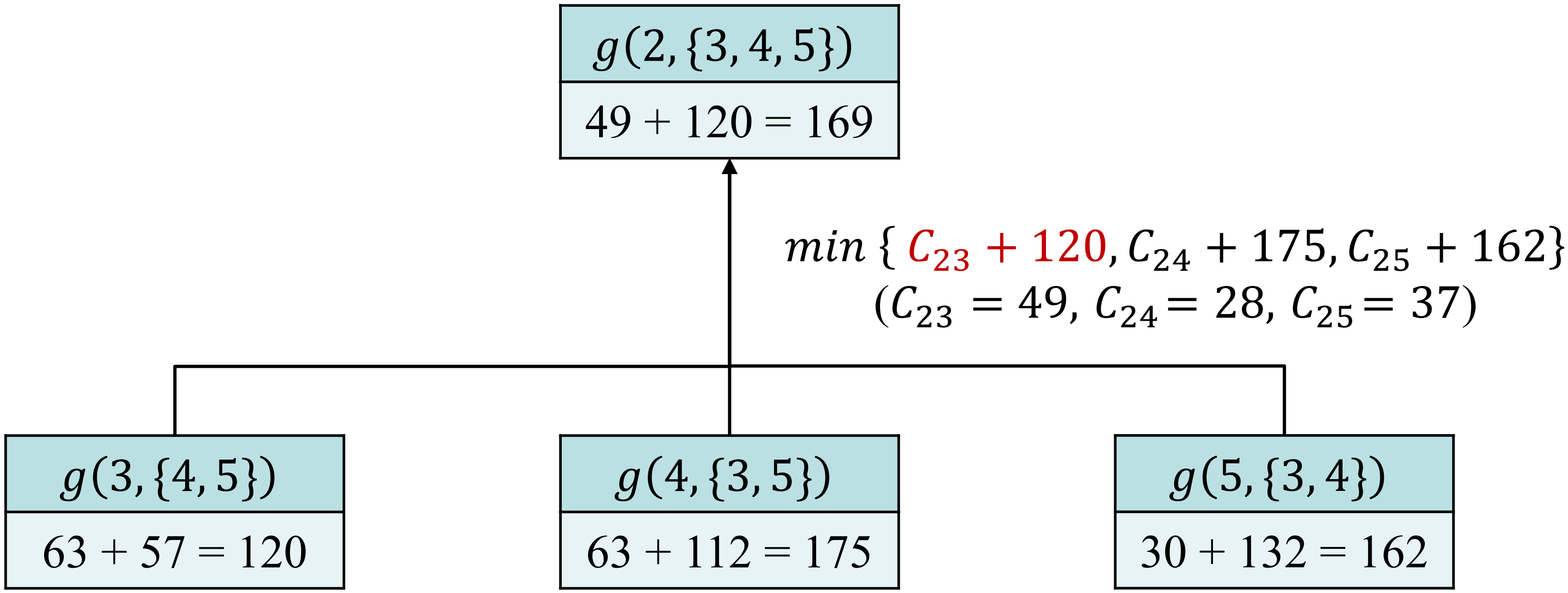
DP: Traveling Salesman Problem

- $i = 5, \; S = \{3, 4\}$



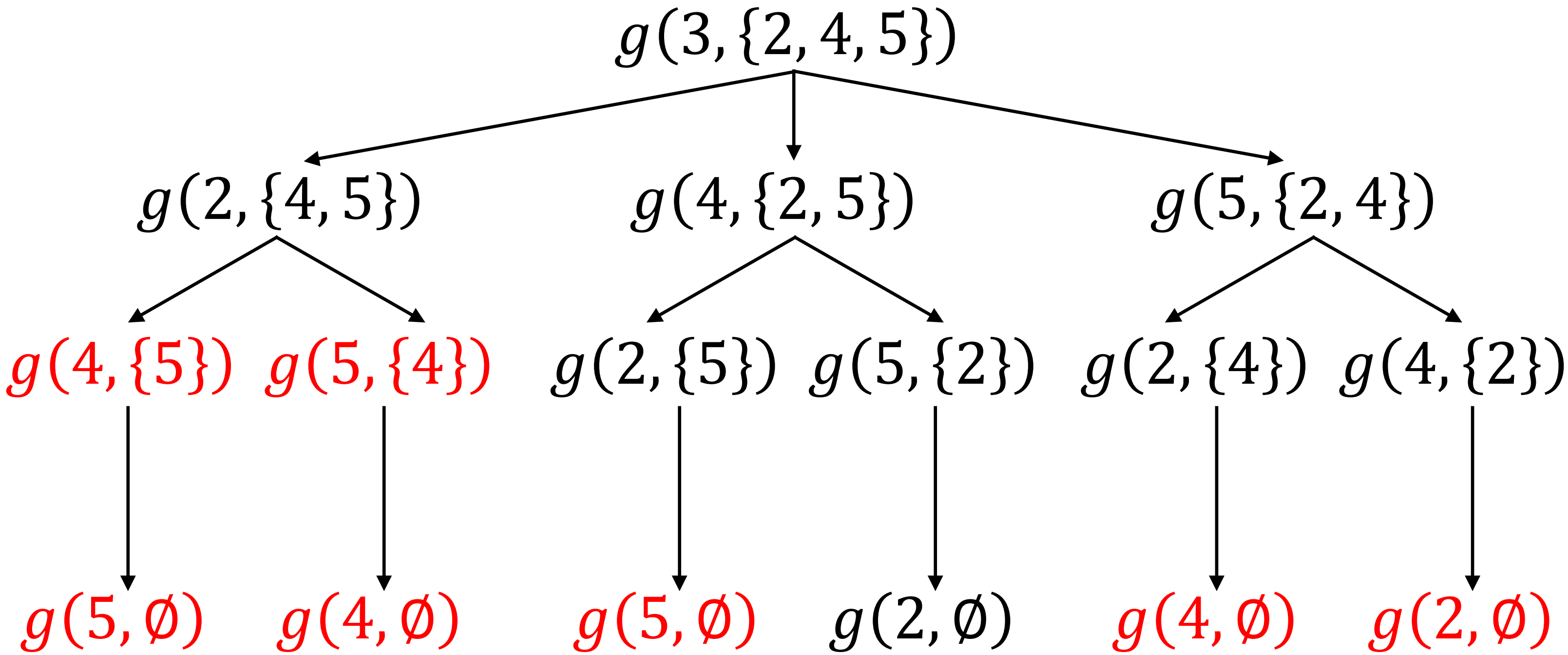
DP: Traveling Salesman Problem

- $i = 2, \ S = \{3, 4, 5\}$



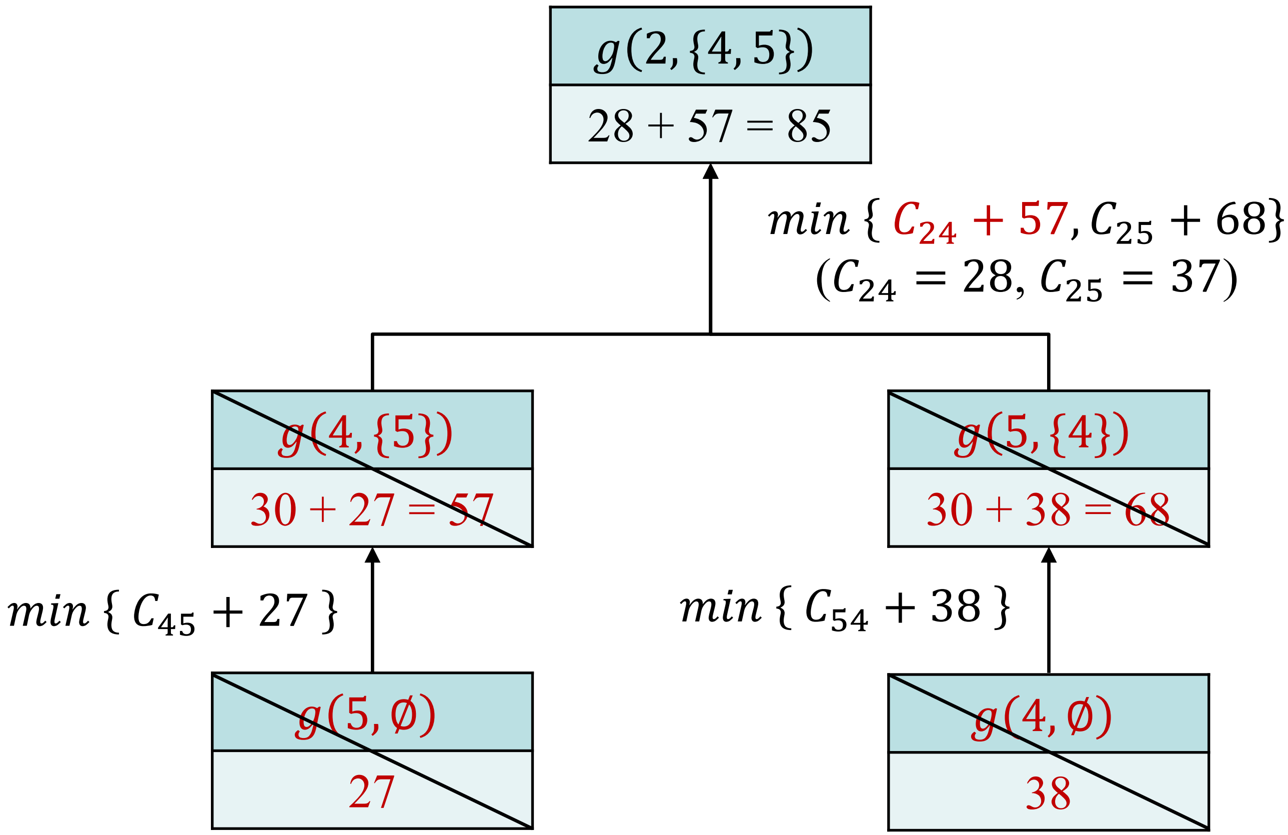
DP: Traveling Salesman Problem

- $i = 3, \; S = \{ 2, 4, 5 \}$



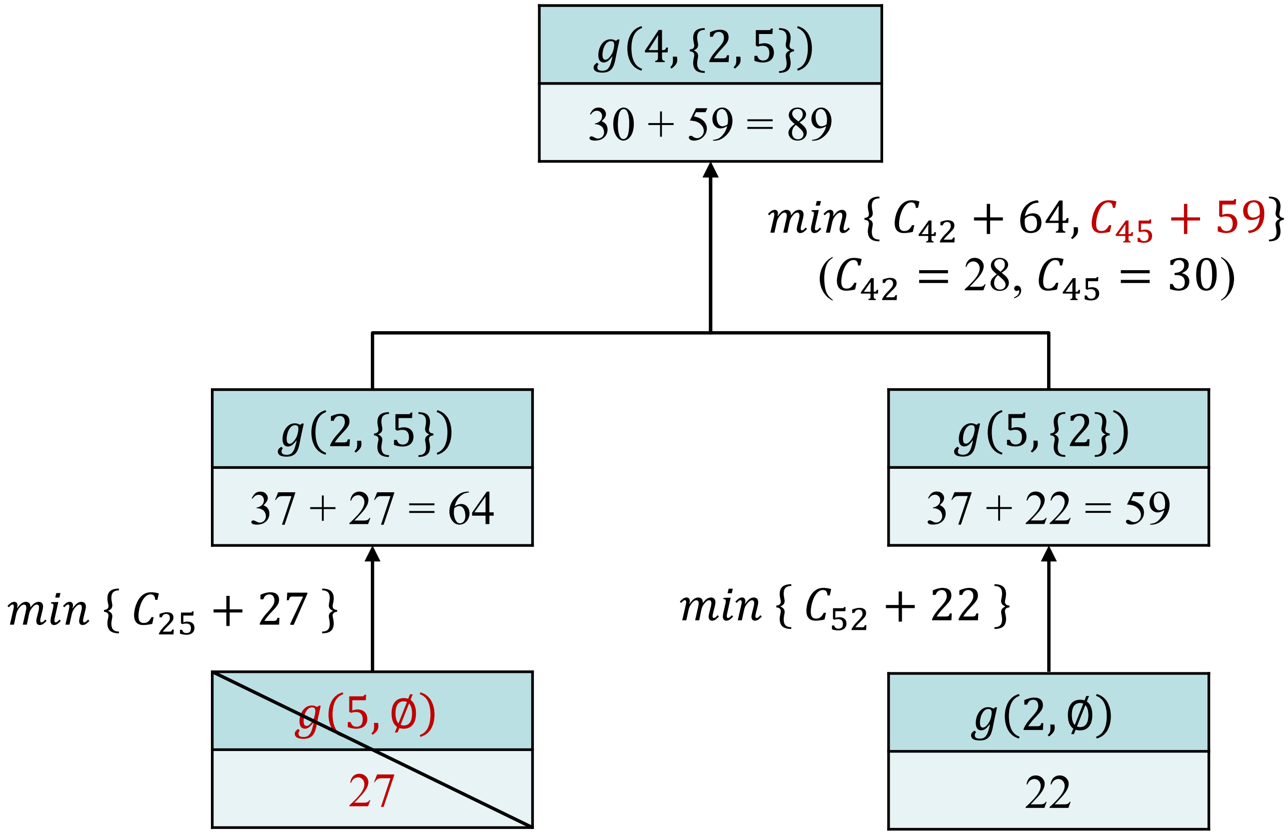
DP: Traveling Salesman Problem

- $i = 2, \; S = \{4, 5\}$



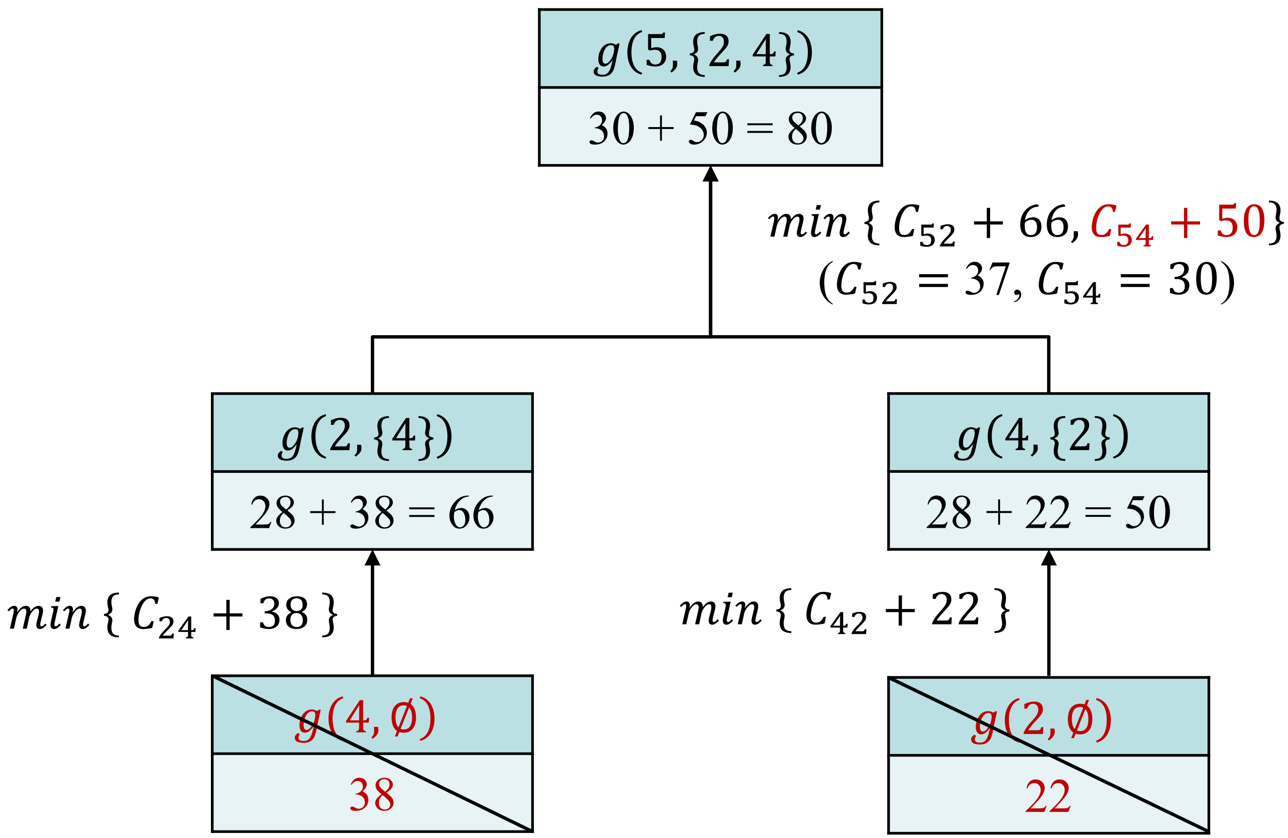
DP: Traveling Salesman Problem

- $i = 4, \;\; S = \{ 2, 5 \}$



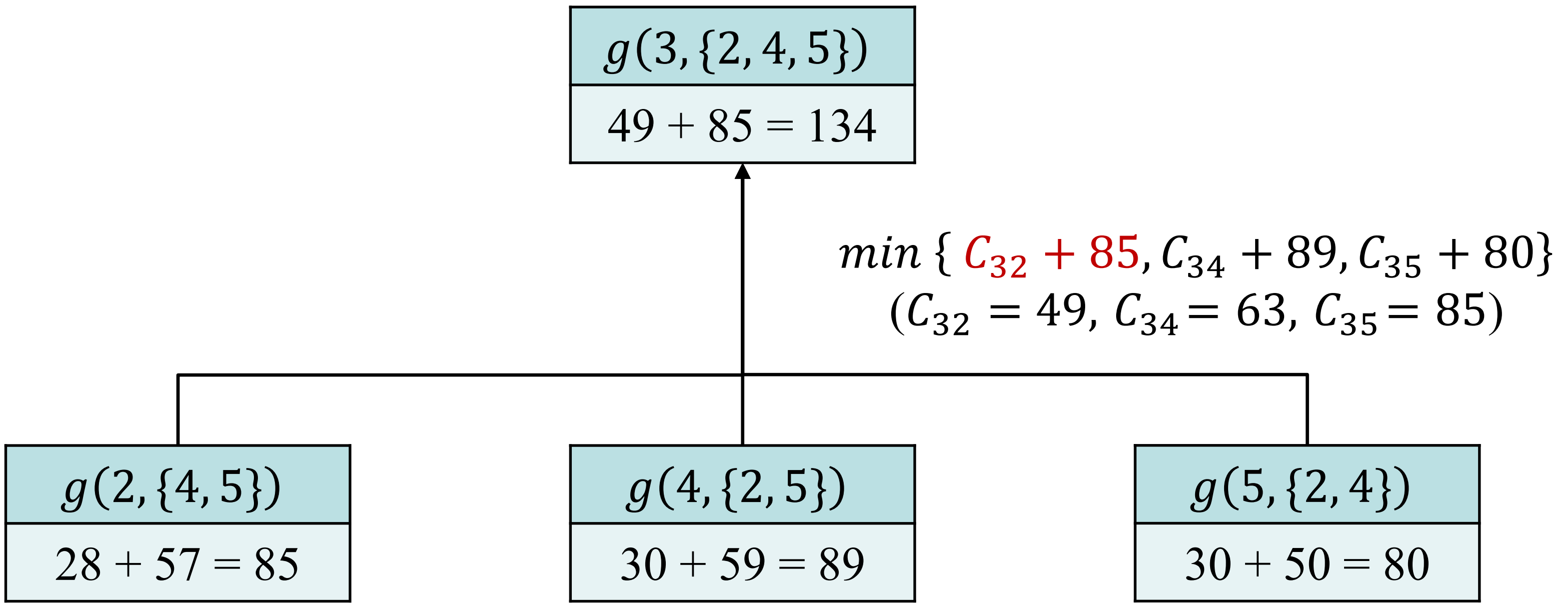
DP: Traveling Salesman Problem

- $i = 5, \ S = \{2, 4\}$



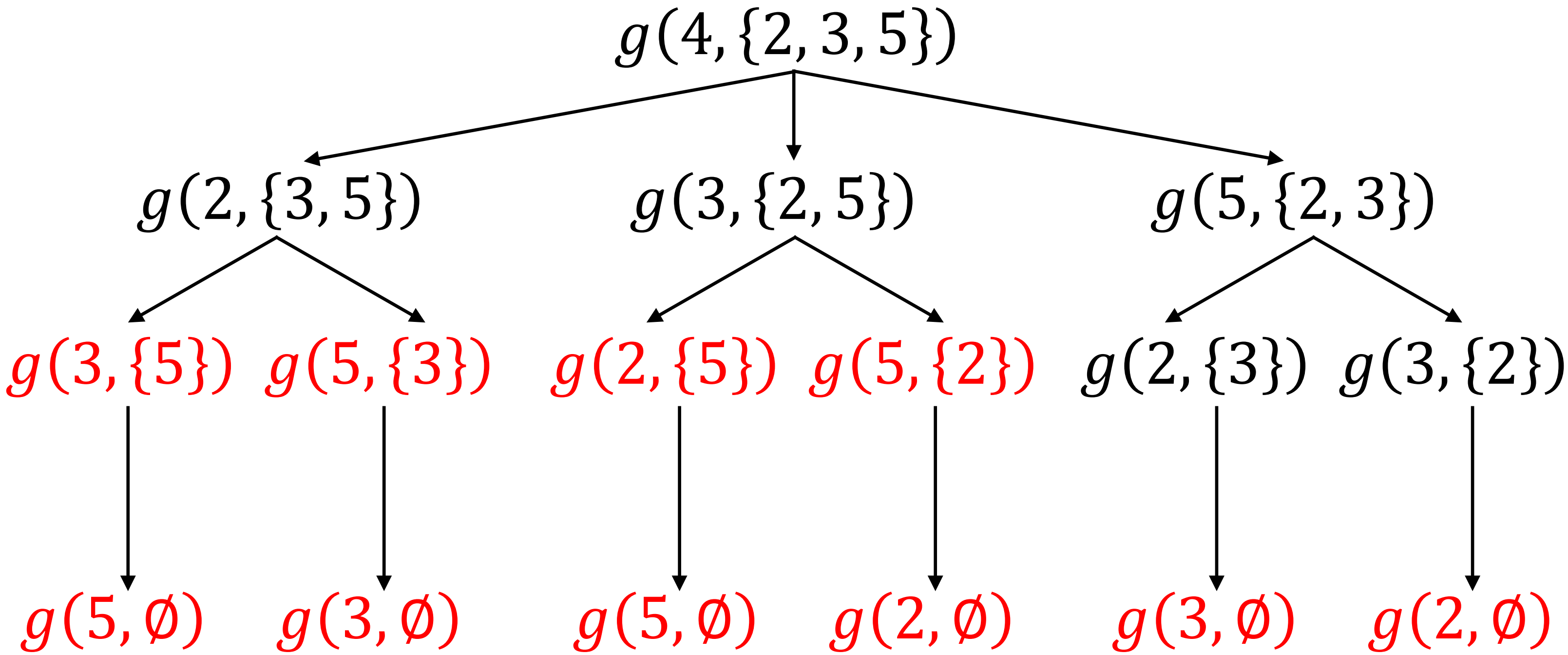
DP: Traveling Salesman Problem

- $i = 3, \ S = \{2, 4, 5\}$



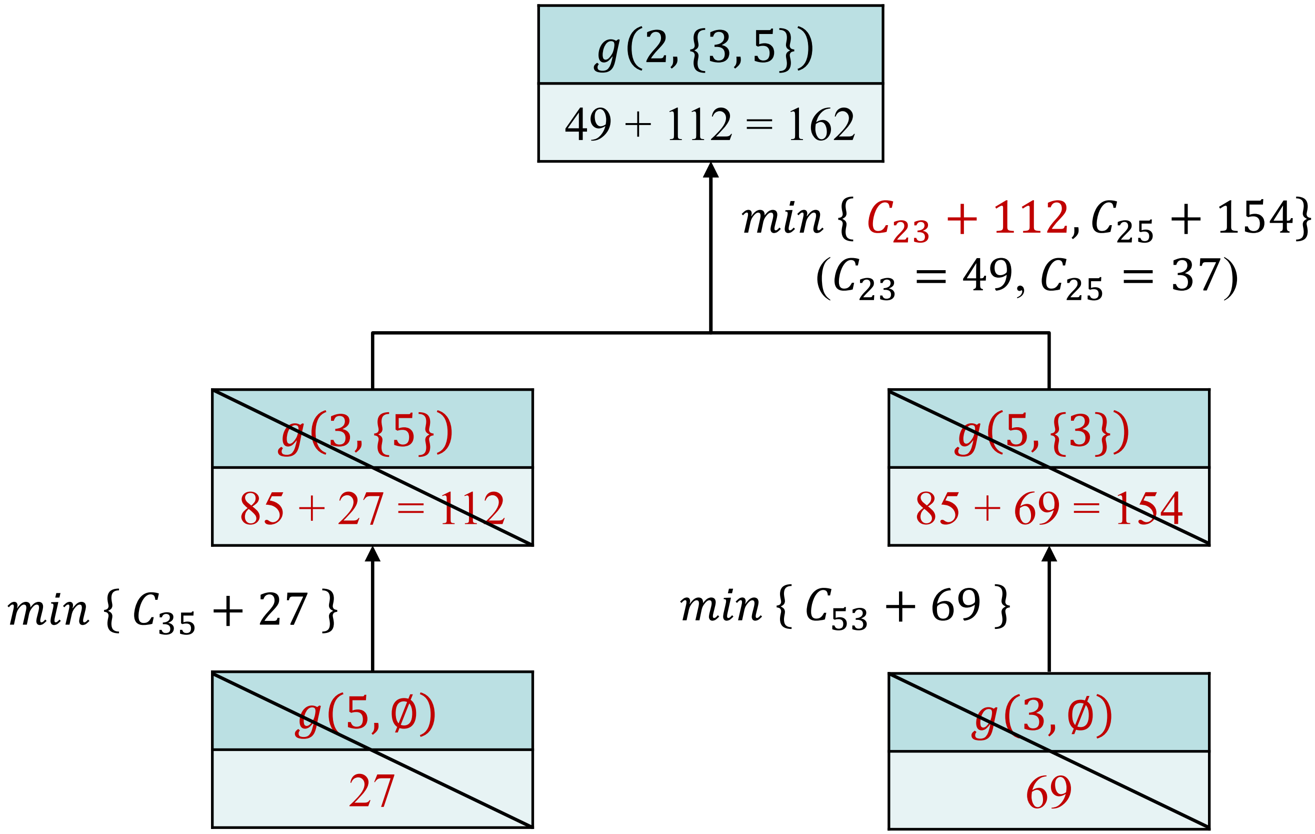
DP: Traveling Salesman Problem

- $i = 4, \; S = \{2, 3, 5\}$



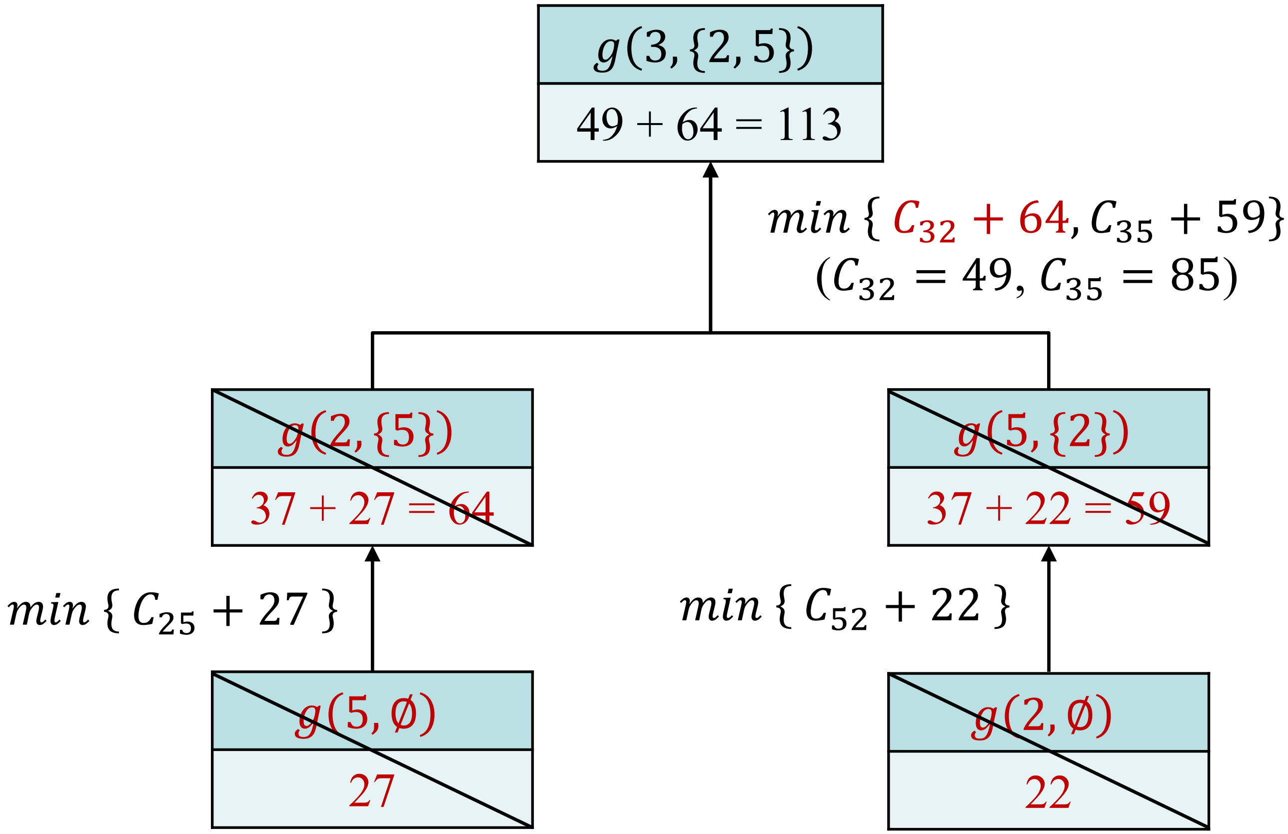
DP: Traveling Salesman Problem

- $i = 2, \quad S = \{3, 5\}$



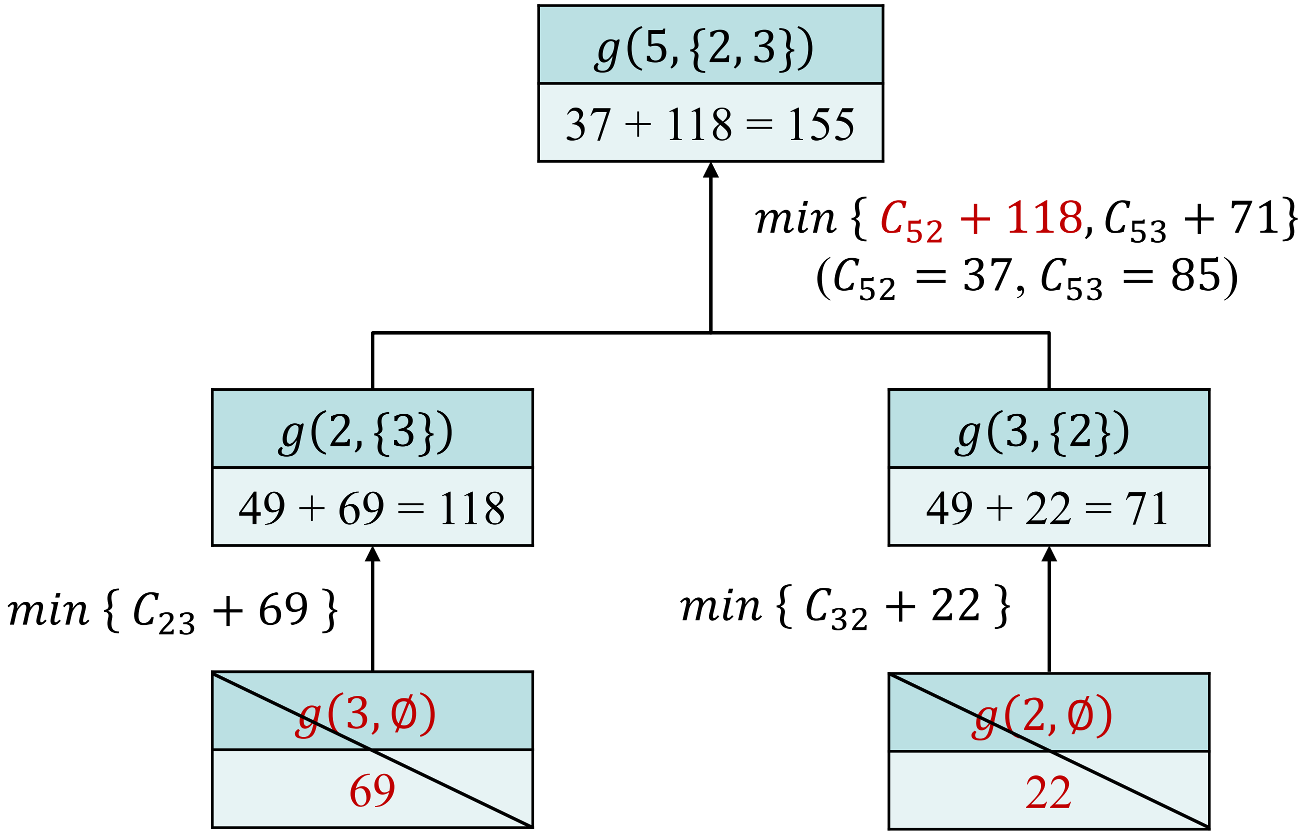
DP: Traveling Salesman Problem

- $i = 3, \;\; S = \{2, 5\}$



DP: Traveling Salesman Problem

- $i = 5, \ S = \{2, 3\}$



DP: Traveling Salesman Problem

- $i\ = 4,\ \ S\ =\ \{\, 2, 3, 5\, \}$

$g(4, \{2, 3, 5\})$

$63 + 113 = 176$

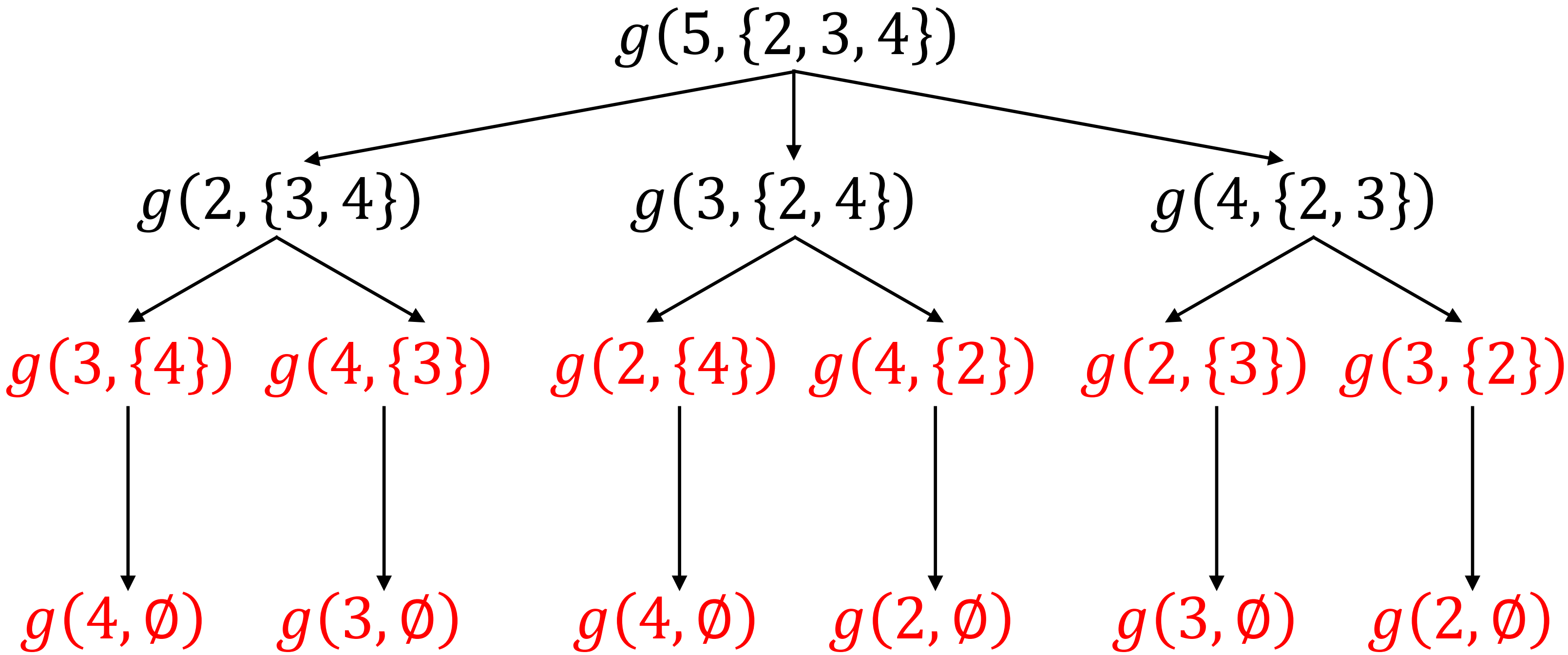
$min\,\{\, C_{42} + 162, C_{43} + 113, C_{45} + 155\}$

$(C_{42}\ = 28,\ C_{43} = 63,\ C_{45} = 30)$

$g(2, \{3, 5\})$	$g(3, \{2, 5\})$	$g(5, \{2, 3\})$
$49 + 112 = 162$	$49 + 64 = 113$	$37 + 118 = 155$

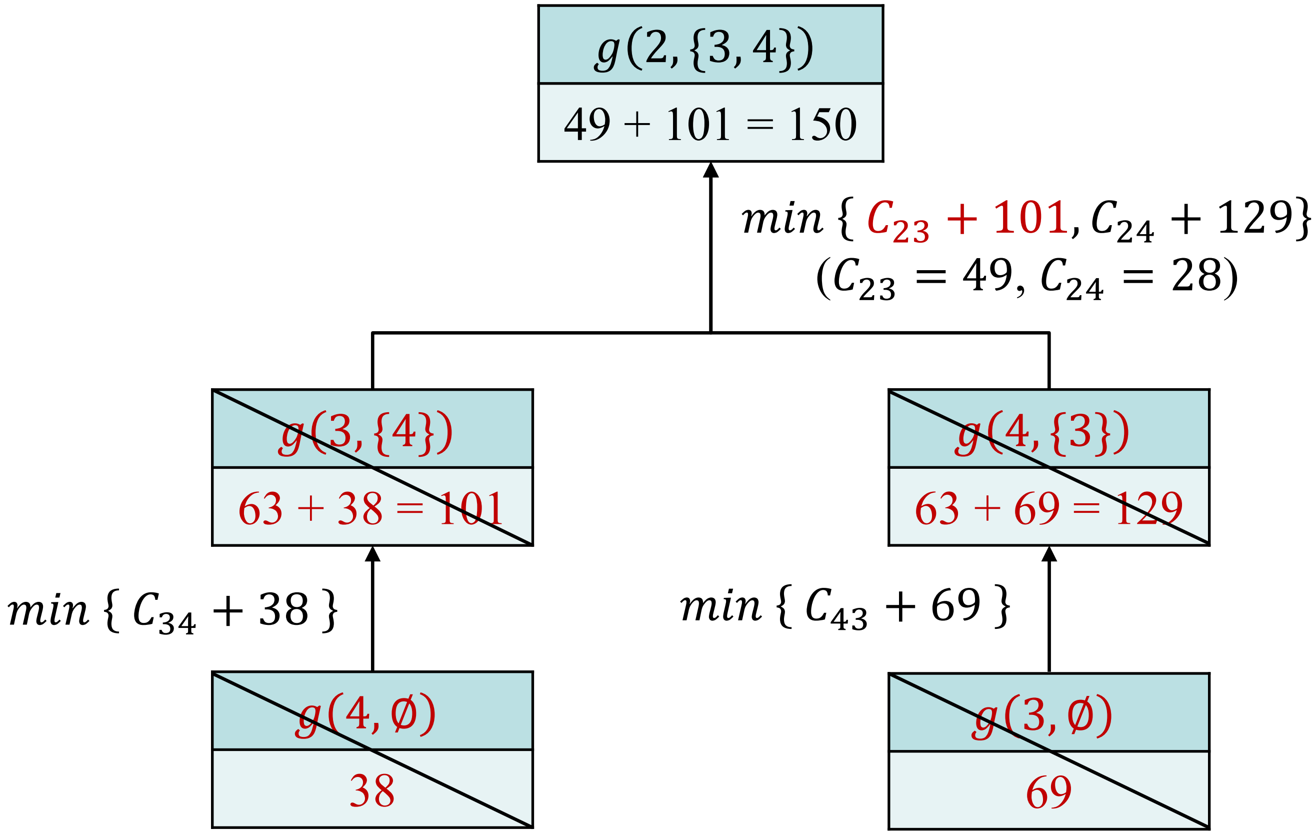
DP: Traveling Salesman Problem

- $i = 5,\ \ S = \{2, 3, 4\}$



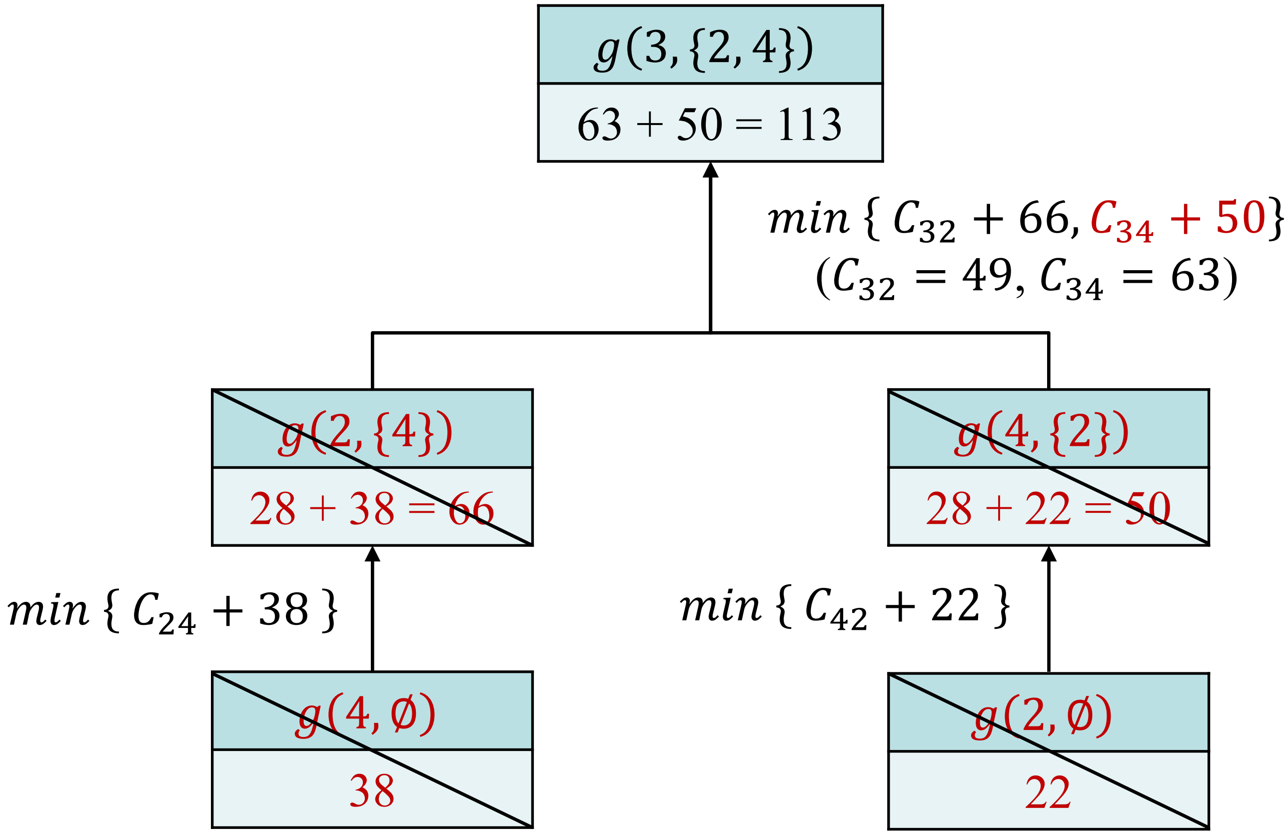
DP: Traveling Salesman Problem

- $i = 2, \; S = \{3, 4\}$



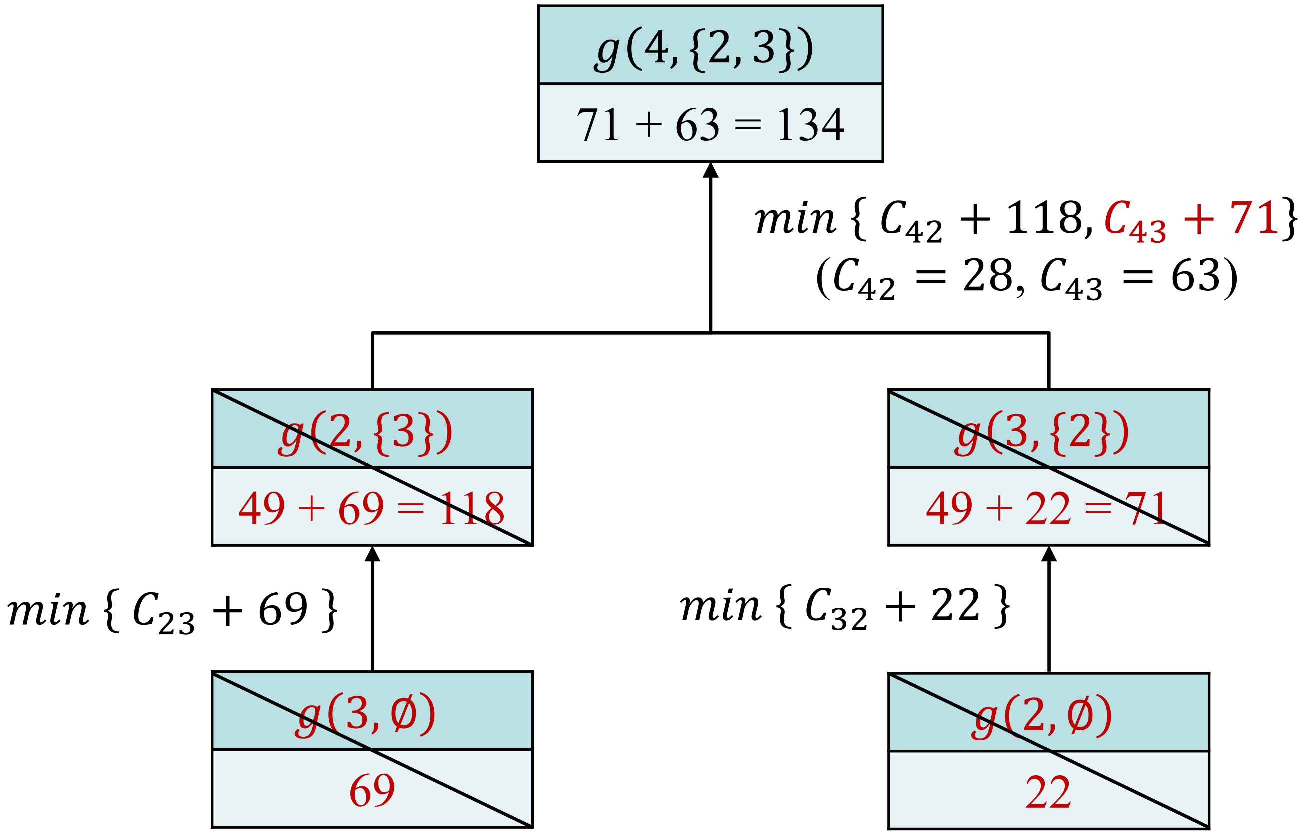
DP: Traveling Salesman Problem

- $i = 3, \ S = \{2, 4\}$



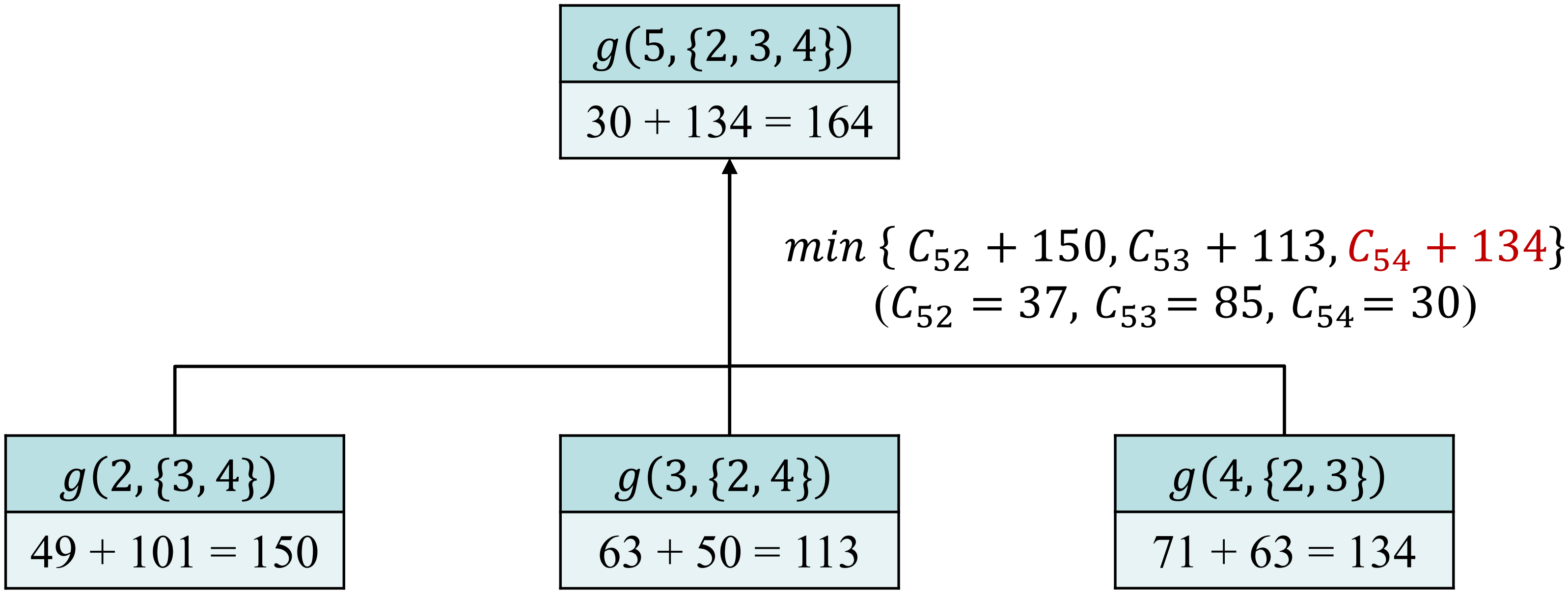
DP: Traveling Salesman Problem

- $i = 4, \; S = \{2, 3\}$



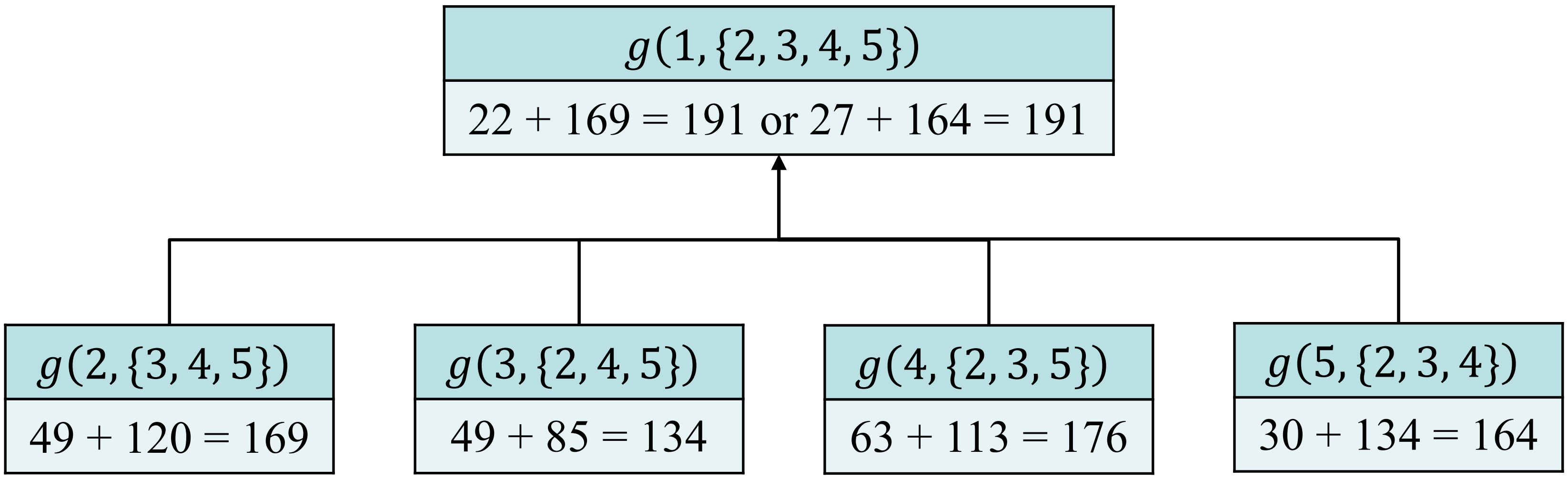
DP: Traveling Salesman Problem

- $i = 5, \ S = \{2, 3, 4\}$



DP: Traveling Salesman Problem

- $i = 1, \quad S = \{2, 3, 4, 5\}$

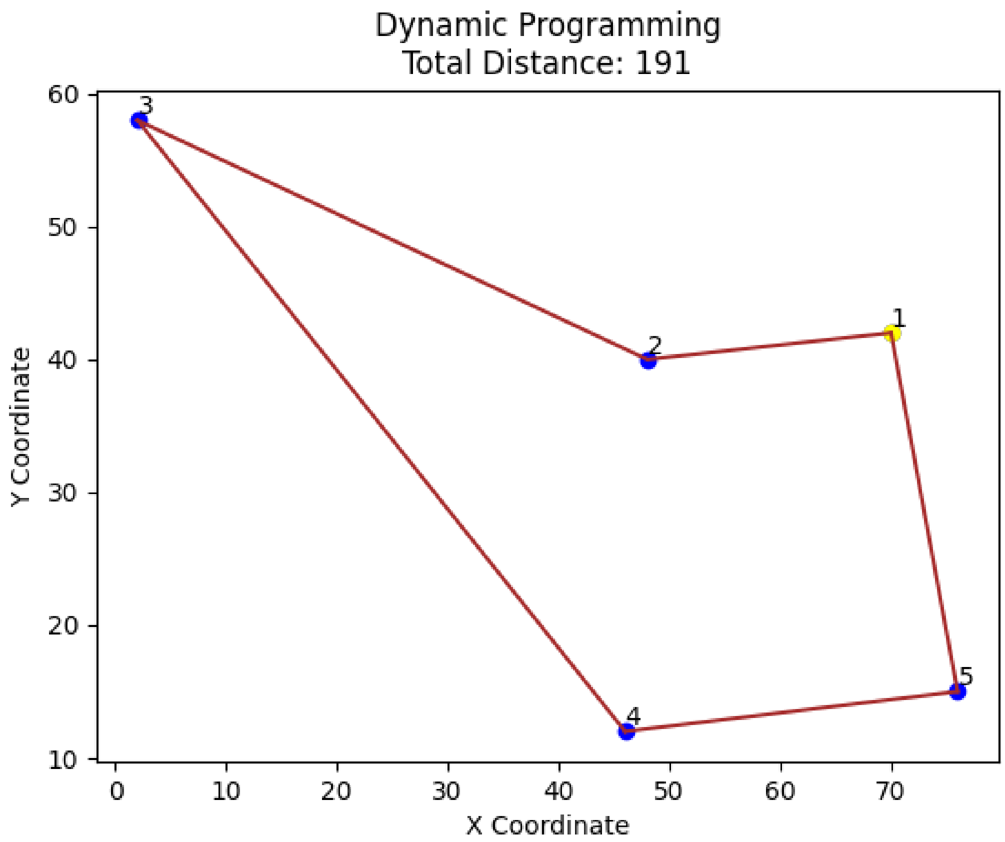


$$min \{ C_{12} + 169, C_{13} + 134, C_{14} + 176, C_{15} + 164 \}$$
$$(C_{12} = 22,\ C_{13} = 69,\ C_{14} = 38,\ C_{15} = 27)$$

- Minimum Cost: 191
- Best Path: $[1, 2, 3, 4, 5]\ or\ [1, 5, 4, 3, 2]$

Traveling Salesman Problem

- Dynamic Programming



Ant Colony Algorithm (ACO)

- Ant colony (群體) optimization (ACO) takes inspiration from the foraging behavior (覓食行為) of some ant species.

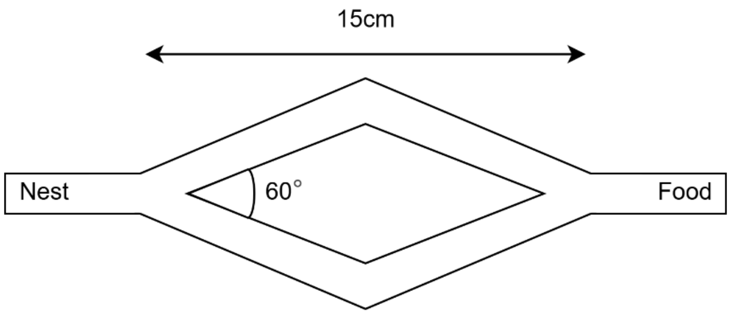
- These ants deposit pheromone (費洛蒙) on the ground in order to mark some favorable path that should be followed by other members of the colony.

Biological Inspiration

- Stigmergy (共識主動性)
  - Stigmergy is an indirect, non-symbolic form of communication mediated by the environment
  - Stigmergic information is local: it can only be accessed by those insects that visit the locus in which it was released

Double Bridge Experiment

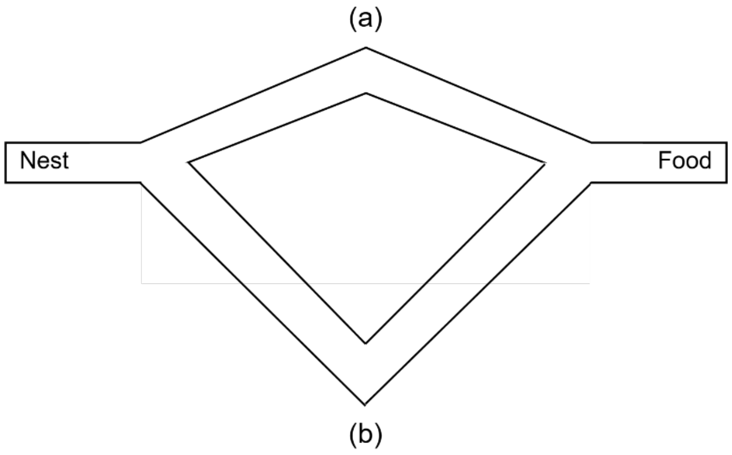
- Branches have equal lengths
  - Each ant randomly chooses one of the two bridges
    - Ants start to explore the surrounding of the nest
    - Ants deposit pheromones along their path



  - One of the two bridges accumulates a higher concentration of pheromones
  - Over time, the entire colony converges to using the same bridge

Double Bridge Experiment

- Branches have different lengths
  - The short bridge is the first to reach the nest
  - Faster pheromone accumulation on the short bridge
  - Higher probability of more ants choosing the short bridge



  - Further ants select the short one instead of the long one

Ant Colony Optimization Algorithms

- Several ACO algorithms have been proposed in the literature.
  - e.g., Ants, Hyper-Cube AS, Rank-Based AS, etc.
- Main ACO Algorithms
  - Ant System (AS)
  - Variants
    - $Max - Min$ Ant System (MMAS)
    - Ant Colony System (ACS)

Ant System (AS)

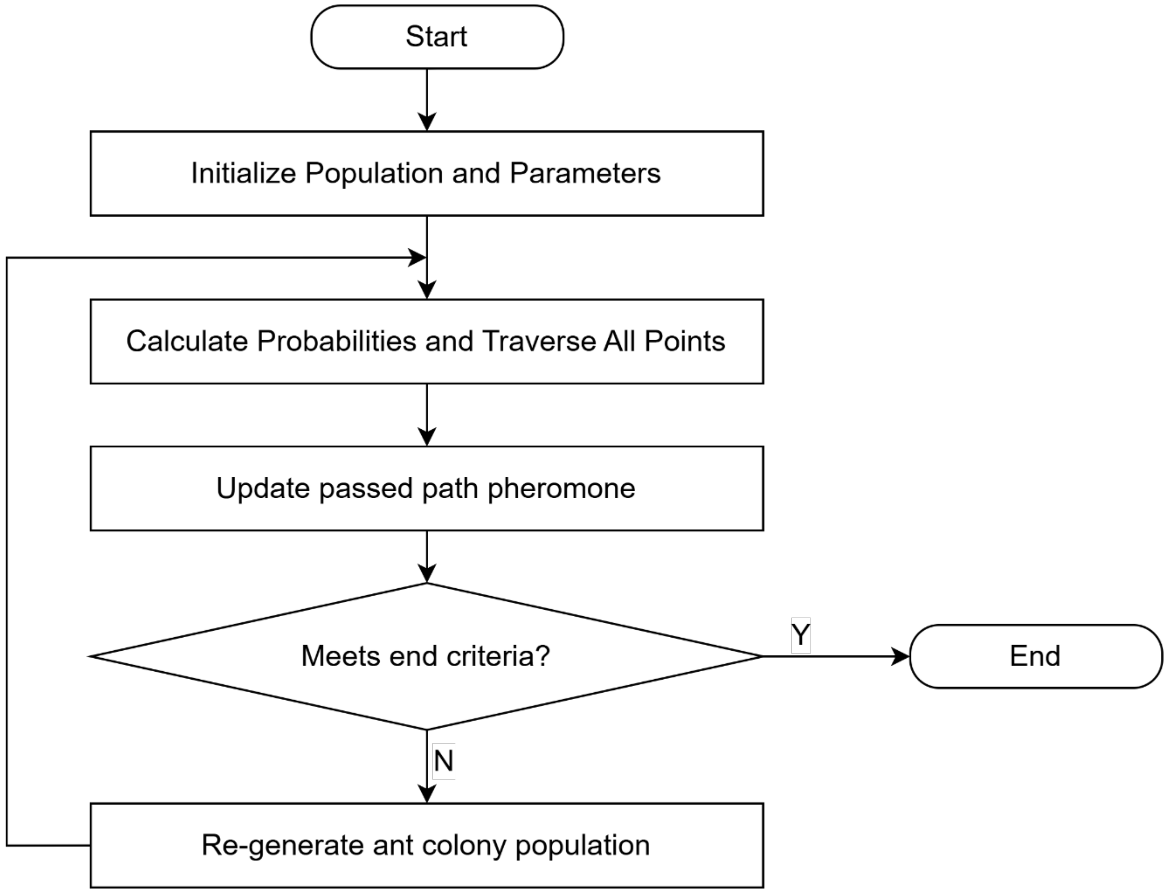
- ACO for Traveling Salesman Problem (TSP)
  - Iterative Algorithm
    - Simulate ants moving on a graph
    - Allow each city to be visited once and only once
    - Ants select the next city stochastically from unvisited cities
  - Pheromone Mechanism
    - Ants can read and modify pheromone
    - Path selection is biased by pheromone concentrations
    - At the end of each iteration, pheromone values are updated to influence future decisions.

Ant System(AS)

- Basic Concept
  - Set Initial Positions
    - Each ant starts from a unique city to avoid local optima
  - Calculate Transition Probabilities
    - Ants select the next city based on calculated transition probabilities
  - Complete the Tour and Calculate Path Length
    - After visiting all cities, the ant calculates the total path length.
  - Update Pheromone
    - Pheromone concentrations are updated based on path quality, length, and evaporation rate

Ant System (AS)

- Flowchart



Ant System (AS)

- Proper Nouns
  - Pheromone (費洛蒙)
  - Evaporation Mechanism (揮發機制)
  - Heuristic Information (啟發訊息)
  - Transition Probability (轉移機率)

Ant System (AS)

- Pheromone
  - The pheromone τ_{ij} associated with the edge joining cities i and j.
  - ρ is the evaporation rate, m is the number of ants, and $\Delta\tau_{ij}^{k}$ is the quantity of pheromone laid on edge(i, j) by ant k.
  - t represents the iteration number in the optimization process.
  - At each iteration, the pheromone values are updated by all the m ants that have built a solution in the iteration itself.

$$\tau_{ij}(t+1) \leftarrow (1-\rho) \cdot \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

Ant System (AS)

- Pheromone
  - Q is a constant related to the quantity of trail laid by ants, and L_k is the length of the tour constructed by ant k

$$\Delta\tau_{ij}^{k} = \begin{cases} Q/L_k & \text{if ant } k \text{ used edge}(i,j) \text{ in its tour,} \\ 0 & \text{otherwise,} \end{cases}$$

Ant System (AS)

- Heuristic Information
  - Heuristic information η_{ij} associated with the edge joining cities i and j which is given by:

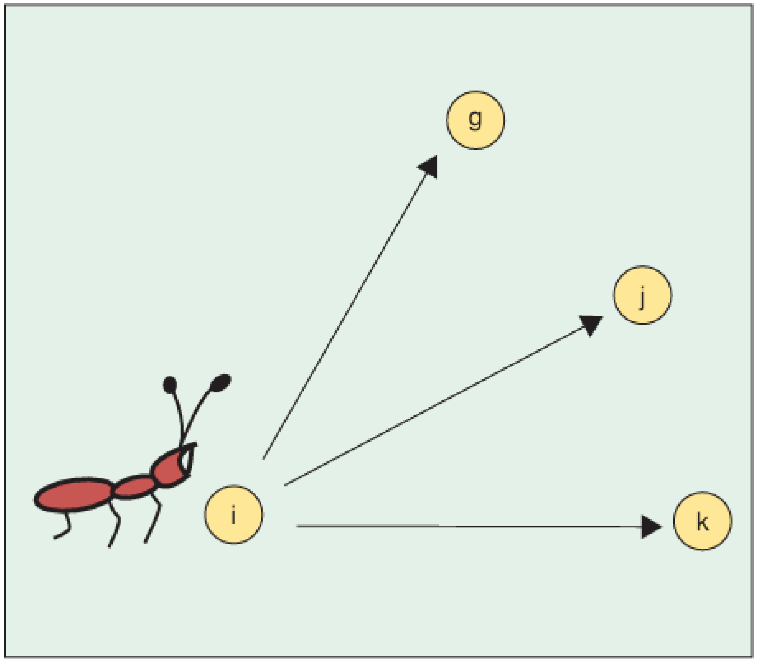
$$\eta_{ij} = \frac{1}{d_{ij}}$$

  - d_{ij} is the distance between cities i and j

$$d_{ij} = \left[(x_i - x_j)^2 + (y_i - y_j)^2\right]^{\frac{1}{2}}$$

Ant System (AS)

- Transition Probability
  - An ant in city i chooses the next city to visit
  - Cities g, j, k has not been previously visited



Ant System (AS)

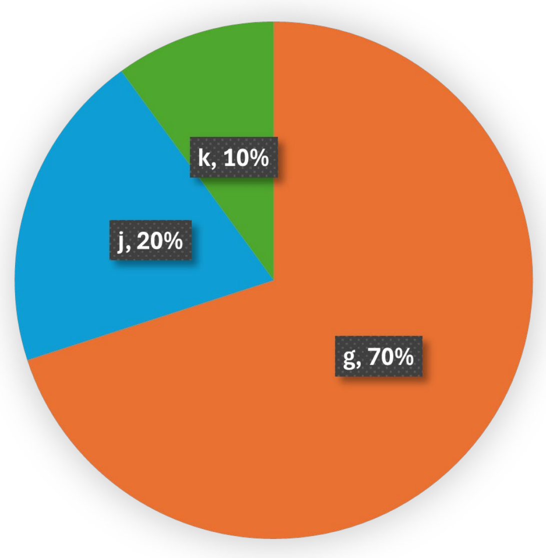
- Transition Probability

$$p_{ij}^k = \begin{cases} \dfrac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{c_{il} \in N(s^p)} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}} & \text{if } c_{ij} \in N(s^p), \\ 0 & \text{otherwise,} \end{cases}$$

  - $N(s^p)$ is the set of feasible components; that is, l is a city not yet visited by the ant k
  - Ant k is in city i and has so far constructed the partial solution s^p , the probability of going to city j is given by:
  - The parameters α and β control the relative importance of the pheromone

Ant System (AS)

- Roulette Wheel
  - Cities with higher transition probabilities have a greater chance of being selected
  - The next city to visit is determined through a random selection process



Ant System (AS)

- Stochastic Mechanism
  - Python random.choices() in C language

```
int main() {
   srand(time(NULL));
   const int items[] = {1, 2, 3, 4};
   double probabilities[] = {0.5, 0.3, 0.1, 0.1};

   int n = 4;  // number of items
   int k = 1;  // number of items to select

   int selected_items[k];

   choices(items, probabilities, n, k, selected_items);

   return 0;
}
```

Ant System (AS)

- Select k random elements based on probabilities
  - normalize()
  - accumulate_probabilities()
  - choose_index()

```
void choices(const int population[], const double probabilities[], int n, int k, int result[]) {
   double cum_probabilities[n];
   double normalized_probabilities[n];
   normalize(probabilities, normalized_probabilities, n);
   accumulate_probabilities(normalized_probabilities, cum_probabilities, n);

   for (int i = 0; i < k; i++) {
      result[i] = population[choose_index(cum_probabilities, n)];
   }
}
```

Ant System (AS)

- normalize()
  - Calculate the total sum of the probabilities
  - Normalize the probabilities

```
void normalize(const double probabilities[], double normalized_probabilities[], int size) {
    double total_probability = 0.0;

    //  Calculate the total sum of the probabilities
    for (int i = 0; i < size; i++) {
        total_probability += probabilities[i];
    }

    // Normalize the probabilities
    for (int i = 0; i < size; i++) {
        normalized_probabilities[i] = probabilities[i] / total_probability;
    }
}
```

Ant System (AS)

- accumulate_probabilities()
  - Initialize the first cumulative probability
  - Accumulate each probability

```
void accumulate_probabilities(const double probabilities[], double cum_probabilities[], int size) {
    // Initialize the first cumulative probability as the first probability value
    cum_probabilities[0] = probabilities[0];

    // Accumulate each probability to the previous cumulative probability
    for (int i = 1; i < size; i++) {
        cum_probabilities[i] = cum_probabilities[i - 1] + probabilities[i];
    }
}
```

Ant System (AS)

- Choose_index()
  - Generate a random number between 0 and 1
  - Find the Selected Index

```
int choose_index(const double cum_probabilities[], int size) {
    // Generate a random number between 0 and 1
    double r = ((double) rand() / RAND_MAX);

    // Find the first cumulative probability that is greater than the random number
    for (int i = 0; i < size; i++) {
        if (r < cum_probabilities[i]) {
            return i;  // return the selected index
        }
    }
    return size - 1;
}
```

Variants: $MMAS$

- $Max - Min$ Ant System ($MMAS$)
  - The value of the pheromone is bound
  - Only best ant updates the pheromone trails
  - τ_{max} and τ_{min} are respectively the upper and lower bounds imposed on the pheromone

$$\tau_{ij}(t+1) \leftarrow \left[(1-\rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij}^{best}\right]_{\tau_{min}}^{\tau_{max}}$$

  - The operator $[x]_b^a$ is defined as:

$$[x]_b^a = \begin{cases} a & \text{if } x > a, \\ b & \text{if } x < b, \\ x & \text{otherwise;} \end{cases}$$

Variants: $MMAS$

- $Max - Min$ Ant System ($MMAS$)

$$\Delta\tau_{ij}^{best} = \begin{cases} \frac{1}{L_{best}} & \text{if } (i,j) \text{ belongs to the best tour,} \\ x & \text{otherwise,} \end{cases}$$

  - L_{best} is the length of the tour of the best ant.
  - L_{best} may be L_{ib} – *iteration-best*, L_{bs} – *best-so-far* or a combination of both
  - L_{ib} is the best tour found in the current iteration
  - L_{bs} is the best solution found since the start of the algorithm

Variants: ACS

- Ant Colony System (ACS)
  - Local pheromone update
    - Performed by all the ants after each construction step
    - s represents a step in the solution construction process
    - Each ant applies it only to the last edge traversed

$$\tau_{ij}(s+1) = (1-\varphi)\cdot\tau_{ij}(s) + \varphi\cdot\tau_0 \quad , \quad \varphi \in (0,1]$$

    - φ is the pheromone decay coefficient
    - τ_0 is the initial value of the pheromone

Variants: ACS

- Ant Colony System (ACS)
  - Offline pheromone update
    - Similarly to $MMAS$ is applied at the end of iteration by only one ant
    - L_{best} can be either the L_{ib} or the L_{bs}

$$\tau_{ij}(t+1) \leftarrow \begin{cases} (1-\rho)\cdot\tau_{ij}(t) \ + \rho\cdot\Delta\tau_{ij} & \text{if } (i,j) \text{ belongs to best tour,} \\ (1-\rho)\cdot\tau_{ij}(t) & \text{otherwise,} \end{cases}$$

  - Pseudorandom Proportional rule
    - The probability for an ant to move from city i to city j depends on a random variable q uniformly distributed over $[0, 1]$
    - if $q \ \leq \ q0$, then $j \ = \ \arg max_{c_{il}\in \mathrm{N}(s^p)} \left\{\tau_{il}\eta_{il}^{\beta}\right\}$, otherwise original Equation is used

ACO: Traveling Salesman Problem

- City Coordinates

1	2	3	4	5
(70, 42)	(48, 40)	(2, 58)	(46, 12)	(76, 15)

- Distance d_{ij} between any two cities i and j
  - $d_{ij} = Round(d_{ij}, 0)$

	1	2	3	4	5
1	0	22	69	38	27
2	22	0	49	28	37
3	69	49	0	63	85
4	38	28	63	0	30
5	27	37	85	30	0

ACO: Traveling Salesman Problem

- Parameters
  - $\alpha = 1, \ \beta = 1, \ \rho = 0.5, \ Q = 100, \ Start\ City = 1$

- Pheromone Matrix
  - For every edge(i, j) set an initial value $\tau_{ij} = 1$

	1	2	3	4	5
1	0	1	1	1	1
2	1	0	1	1	1
3	1	1	0	1	1
4	1	1	1	0	1
5	1	1	1	1	0

ACO: Traveling Salesman Problem

- $Iteration = 1$
  - $k = 1,\ s^p = \{\,1\,\},\ N(s^p) = \{\,2, 3, 4, 5\,\},\ Selected = C_{1\to 5}$

$\boldsymbol{C_{i\to j}}$	$\boldsymbol{d_{ij}}$	$\boldsymbol{\tau_{ij}}$	$\boldsymbol{P_{ij}^{k}}$
$C_{1\to 2}$	22	1	$\dfrac{1^1\cdot\left(\frac{1}{22}\right)^1}{1^1\cdot\left(\frac{1}{22}\right)^1+1^1\cdot\left(\frac{1}{69}\right)^1+1^1\cdot\left(\frac{1}{38}\right)^1+1^1\cdot\left(\frac{1}{27}\right)^1}=0.37$
$C_{1\to 3}$	69	1	$\dfrac{1^1\cdot\left(\frac{1}{69}\right)^1}{1^1\cdot\left(\frac{1}{22}\right)^1+1^1\cdot\left(\frac{1}{69}\right)^1+1^1\cdot\left(\frac{1}{38}\right)^1+1^1\cdot\left(\frac{1}{27}\right)^1}=0.12$
$C_{1\to 4}$	38	1	$\dfrac{1^1\cdot\left(\frac{1}{38}\right)^1}{1^1\cdot\left(\frac{1}{22}\right)^1+1^1\cdot\left(\frac{1}{69}\right)^1+1^1\cdot\left(\frac{1}{38}\right)^1+1^1\cdot\left(\frac{1}{27}\right)^1}=0.21$
$C_{1\to 5}$	27	1	$\dfrac{1^1\cdot\left(\frac{1}{27}\right)^1}{1^1\cdot\left(\frac{1}{22}\right)^1+1^1\cdot\left(\frac{1}{69}\right)^1+1^1\cdot\left(\frac{1}{38}\right)^1+1^1\cdot\left(\frac{1}{27}\right)^1}=0.30$

ACO: Traveling Salesman Problem

- $Iteration = 1$
  - $k = 1,\ s^p = \{1,5\},\ N(s^p) = \{2,3,4\},\ Selected = C_{5\to 2}$

$\boldsymbol{C_{i\to j}}$	$\boldsymbol{d_{ij}}$	$\boldsymbol{\tau_{ij}}$	$\boldsymbol{P_{ij}^{k}}$
$C_{5\to 2}$	37	1	$\dfrac{1^1\cdot\left(\frac{1}{37}\right)^1}{1^1\cdot\left(\frac{1}{37}\right)^1+1^1\cdot\left(\frac{1}{85}\right)^1+1^1\cdot\left(\frac{1}{30}\right)^1}=0.38$
$C_{5\to 3}$	85	1	$\dfrac{1^1\cdot\left(\frac{1}{85}\right)^1}{1^1\cdot\left(\frac{1}{37}\right)^1+1^1\cdot\left(\frac{1}{85}\right)^1+1^1\cdot\left(\frac{1}{30}\right)^1}=0.16$
$C_{5\to 4}$	30	1	$\dfrac{1^1\cdot\left(\frac{1}{30}\right)^1}{1^1\cdot\left(\frac{1}{37}\right)^1+1^1\cdot\left(\frac{1}{85}\right)^1+1^1\cdot\left(\frac{1}{30}\right)^1}=0.46$

ACO: Traveling Salesman Problem

- $Iteration = 1$
  - $k = 1,\ s^p = \{\,1,5,2\,\},\ N(s^p) = \{\,3,4\,\}\,,\ Selected = C_{2\to4}$

$C_{i\to j}$	d_{ij}	τ_{ij}	P_{ij}^{k}
$C_{2\to3}$	49	1	$\dfrac{1^1\cdot\left(\frac{1}{49}\right)^1}{1^1\cdot\left(\frac{1}{49}\right)^1+1^1\cdot\left(\frac{1}{28}\right)^1}=0.36$
$C_{2\to4}$	28	1	$\dfrac{1^1\cdot\left(\frac{1}{28}\right)^1}{1^1\cdot\left(\frac{1}{49}\right)^1+1^1\cdot\left(\frac{1}{28}\right)^1}=0.64$

ACO: Traveling Salesman Problem

- $Iteration = 1$
  - $k = 1,\ s^p = \{\,1,5,2,4\,\},\ N(s^p) = \{\,3\,\},\ Selected = C_{4\to3}$

$C_{i\to j}$	d_{ij}	τ_{ij}	P_{ij}^{k}
$C_{4\to3}$	63	1	$\dfrac{1^1\cdot\left(\frac{1}{63}\right)^1}{1^1\cdot\left(\frac{1}{63}\right)^1} = 1$

  - $k = 1,\ s^p = \{\,1,5,2,4,3\,\},\ N(s^p) = \emptyset$
    - $s = \{\,1,5,2,4,3\,\}$
    - $L_1 = 27 + 37 + 28 + 63 + 69 = 224,$
    - $\Delta\tau_{ij}^{1} = \dfrac{Q}{L_1} = \dfrac{100}{224} = 0.45$

ACO: Traveling Salesman Problem

- $Iteration = 1$
  - $k = 1,\ s^p = \{1,5,2,4,3\},\ N(s^p) = \emptyset$
    - $s = \{1,5,2,4,3\},\ L_1 = 224,\ \Delta\tau_{ij}^1 = \frac{Q}{L_1} = \frac{100}{224} = 0.45$
  - $k = 2,\ s^p = \{2,5,1,4,3\},\ N(s^p) = \emptyset$
    - $s = \{1,4,3,2,5\},\ L_2 = 214,\ \Delta\tau_{ij}^2 = \frac{Q}{L_2} = \frac{100}{214} = 0.47$
  - $k = 3,\ s^p = \{3,5,2,4,1\},\ N(s^p) = \emptyset$
    - $s = \{1,3,5,2,4\},\ L_3 = 257,\ \Delta\tau_{ij}^3 = \frac{Q}{L_3} = \frac{100}{257} = 0.39$
  - $k = 4,\ s^p = \{4,5,2,1\ 3\},\ N(s^p) = \emptyset$
    - $s = \{1,3,4,5,2\},\ L_4 = 221,\ \Delta\tau_{ij}^4 = \frac{Q}{L_4} = \frac{100}{221} = 0.45$
  - $k = 5,\ s^p = \{5,1,2,4,3\},\ N(s^p) = \emptyset$
    - $s = \{1,2,4,3,5\},\ L_5 = 225,\ \Delta\tau_{ij}^5 = \frac{Q}{L_5} = \frac{100}{225} = 0.44$

ACO: Traveling Salesman Problem

- $Iteration = 1$
  - $s_{best} = \emptyset,\ L_{best} = inf$

$\boldsymbol{ant_k}$	$\boldsymbol{L_k}$	$\boldsymbol{s_k}$	$\boldsymbol{L_{best}}$	$\boldsymbol{s_{best}}$
ant_1	224	$\{1, 5, 2, 4, 3\}$	224	$\{1, 5, 2, 4, 3\}$
ant_2	214	$\{1, 4, 3, 2, 5\}$	214	$\{1, 4, 3, 2, 5\}$
ant_3	257	$\{1, 3, 5, 2, 4\}$	214	$\{1, 4, 3, 2, 5\}$
ant_4	221	$\{1, 3, 4, 5, 2\}$	214	$\{1, 4, 3, 2, 5\}$
ant_5	225	$\{1, 2, 4, 3, 5\}$	214	$\{1, 4, 3, 2, 5\}$

ACO: Traveling Salesman Problem

- $Iteration = 1$
  - Pheromone Evaporation

	1	**2**	**3**	**4**	**5**
1	0	$1 \cdot (1\text{-}\rho) = 0.5$	$1 \cdot (1\text{-}\rho) = 0.5$	$1 \cdot (1\text{-}\rho) = 0.5$	$1 \cdot (1\text{-}\rho) = 0.5$
2	$1 \cdot (1\text{-}\rho) = 0.5$	0	$1 \cdot (1\text{-}\rho) = 0.5$	$1 \cdot (1\text{-}\rho) = 0.5$	$1 \cdot (1\text{-}\rho) = 0.5$
3	$1 \cdot (1\text{-}\rho) = 0.5$	$1 \cdot (1\text{-}\rho) = 0.5$	0	$1 \cdot (1\text{-}\rho) = 0.5$	$1 \cdot (1\text{-}\rho) = 0.5$
4	$1 \cdot (1\text{-}\rho) = 0.5$	$1 \cdot (1\text{-}\rho) = 0.5$	$1 \cdot (1\text{-}\rho) = 0.5$	0	$1 \cdot (1\text{-}\rho) = 0.5$
5	$1 \cdot (1\text{-}\rho) = 0.5$	$1 \cdot (1\text{-}\rho) = 0.5$	$1 \cdot (1\text{-}\rho) = 0.5$	$1 \cdot (1\text{-}\rho) = 0.5$	0

ACO: Traveling Salesman Problem

- $Iteration = 1$
  - $k = 1,\ s = \{\,1, 5, 2, 4, 3\,\},\ L_1 = 224,\ \Delta\tau_{ij}^{1} = 0.45$

	1	2	3	4	5
1	0	0.5	$0.5 + \Delta\tau_{ij}^{1}$	0.5	$0.5 + \Delta\tau_{ij}^{1}$
2	0.5	0	0.5	$0.5 + \Delta\tau_{ij}^{1}$	$0.5 + \Delta\tau_{ij}^{1}$
3	$0.5 + \Delta\tau_{ij}^{1}$	0.5	0	$0.5 + \Delta\tau_{ij}^{1}$	0.5
4	0.5	$0.5 + \Delta\tau_{ij}^{1}$	$0.5 + \Delta\tau_{ij}^{1}$	0	0.5
5	$0.5 + \Delta\tau_{ij}^{1}$	$0.5 + \Delta\tau_{ij}^{1}$	0.5	0.5	0

ACO: Traveling Salesman Problem

- $Iteration = 1$
  - $k = 2,\ s = \{1, 4, 3, 2, 5\},\ L_2 = 214,\ \Delta\tau_{ij}^2 = 0.47$

	1	**2**	**3**	**4**	**5**
1	0	0.5	0.95	$0.5 + \Delta\tau_{ij}^2$	$0.95 + \Delta\tau_{ij}^2$
2	0.5	0	$0.5 + \Delta\tau_{ij}^2$	0.95	$0.95 + \Delta\tau_{ij}^2$
3	0.95	$0.5 + \Delta\tau_{ij}^2$	0	$0.95 + \Delta\tau_{ij}^2$	0.5
4	$0.5 + \Delta\tau_{ij}^2$	0.95	$0.95 + \Delta\tau_{ij}^2$	0	0.5
5	$0.95 + \Delta\tau_{ij}^2$	$0.95 + \Delta\tau_{ij}^2$	0.5	0.5	0

ACO: Traveling Salesman Problem

- $Iteration = 1$
  - $k = 3,\ s = \{1, 3, 5, 2, 4\},\ L_3 = 257,\ \Delta\tau_{ij}^{3} = 0.39$

	1	**2**	**3**	**4**	**5**
1	0	0.5	$0.95 + \Delta\tau_{ij}^{3}$	$0.97 + \Delta\tau_{ij}^{3}$	1.42
2	0.5	0	0.97	$0.95 + \Delta\tau_{ij}^{3}$	$1.42 + \Delta\tau_{ij}^{3}$
3	$0.95 + \Delta\tau_{ij}^{3}$	0.97	0	1.42	$0.5 + \Delta\tau_{ij}^{3}$
4	$0.97 + \Delta\tau_{ij}^{3}$	$0.95 + \Delta\tau_{ij}^{3}$	1.42	0	0.5
5	1.42	$1.42 + \Delta\tau_{ij}^{3}$	$0.5 + \Delta\tau_{ij}^{3}$	0.5	0

ACO: Traveling Salesman Problem

- $Iteration = 1$
  - $k = 4,\ s = \{1, 3, 4, 5, 2\},\ L_4 = 221,\ \Delta\tau_{ij}^4 = 0.45$

	1	**2**	**3**	**4**	**5**
1	0	$0.5 + \Delta\tau_{ij}^4$	$1.34 + \Delta\tau_{ij}^4$	1.36	1.42
2	$0.5 + \Delta\tau_{ij}^4$	0	0.97	1.34	$1.81 + \Delta\tau_{ij}^4$
3	$1.34 + \Delta\tau_{ij}^4$	0.97	0	$1.42 + \Delta\tau_{ij}^4$	0.89
4	1.36	1.34	$1.42 + \Delta\tau_{ij}^4$	0	$0.5 + \Delta\tau_{ij}^4$
5	1.42	$1.81 + \Delta\tau_{ij}^4$	0.89	$0.5 + \Delta\tau_{ij}^4$	0

ACO: Traveling Salesman Problem

- $Iteration = 1$
  - $k = 5,\ s = \{1, 2, 4, 3, 5\},\ L_5 = 225,\ \Delta\tau_{ij}^5 = 0.44$

	1	**2**	**3**	**4**	**5**
1	0	$0.95 + \Delta\tau_{ij}^5$	1.79	1.36	$1.42 + \Delta\tau_{ij}^5$
2	$0.95 + \Delta\tau_{ij}^5$	0	0.97	$1.34 + \Delta\tau_{ij}^5$	2.26
3	1.79	0.97	0	$1.87 + \Delta\tau_{ij}^5$	$0.89 + \Delta\tau_{ij}^5$
4	1.36	$1.34 + \Delta\tau_{ij}^5$	$1.87 + \Delta\tau_{ij}^5$	0	0.95
5	$1.42 + \Delta\tau_{ij}^5$	2.26	$0.89 + \Delta\tau_{ij}^5$	0.95	0

ACO: Traveling Salesman Problem

- $Iteration = 1$
  - Pheromone Matrix

	1	**2**	**3**	**4**	**5**
1	0	1.39	1.79	1.36	1.86
2	1.39	0	0.97	1.78	2.26
3	1.79	0.97	0	2.31	1.33
4	1.36	1.78	2.31	0	0.95
5	1.86	2.26	1.33	0.95	0

  - Current Minimum Cost: 214
  - Best Path So Far: $[1, 4, 3, 2, 5]$

ACO: Traveling Salesman Problem

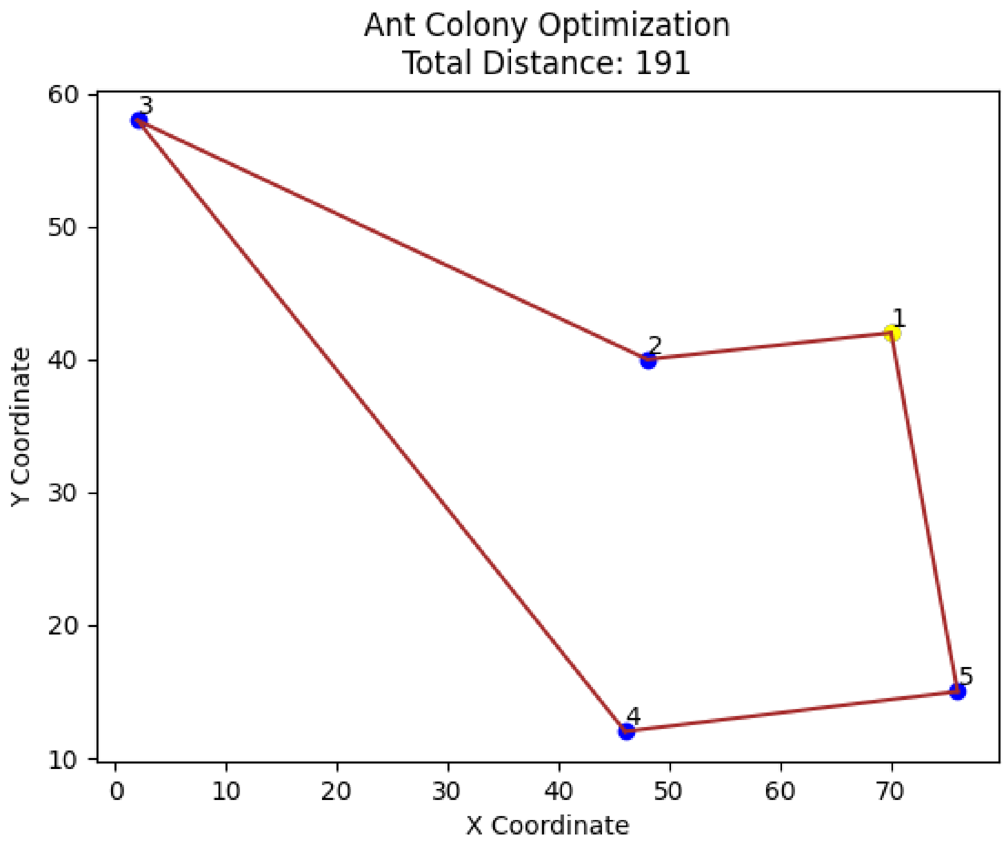
- In the next iteration
  - Replace the current best path with a new one if its cost is lower than the original path
  - Repeat the steps until the end criteria are met
    - The maximum number of iteration is achieved
    - The value repeats more than the allowed number of times

- Result
  - Minimum Cost: 191
  - Best Path: $[1, 2, 3, 4, 5]$ or $[1, 5, 4, 3, 2]$

- ACO do not promise an optimal solution to the problem

ACO: Traveling Salesman Problem

- Ant Colony Optimization



Exercise: Traveling Salesman Problem

- Parameters
  - $\alpha = 1,\ \beta = 1,\ \rho = 0.5,\ Q = 100,\ Start\ City = 1$

- City Coordinates

1	**2**	**3**
(46, 4)	(44, 10)	(32, 97)

- Pheromone Matrix

	1	**2**	**3**
1	0	1	1
2	1	0	1
3	1	1	0

Exercise: Traveling Salesman Problem

- Suppose we have a TSP problem with 3 cities
  - The distance between cities is given by the following matrix

	1	2	3
1	0	6	94
2	6	0	87
3	94	87	0

- What is the min cost and best path in Dynamic Programming and Ant Colony Optimization?
- Compare the differences between these two algorithm