OR 674 DYNAMIC PROGRAMMING

Rajesh Ganesan,
Associate Professor
Systems Engineering and Operations Research
George Mason University

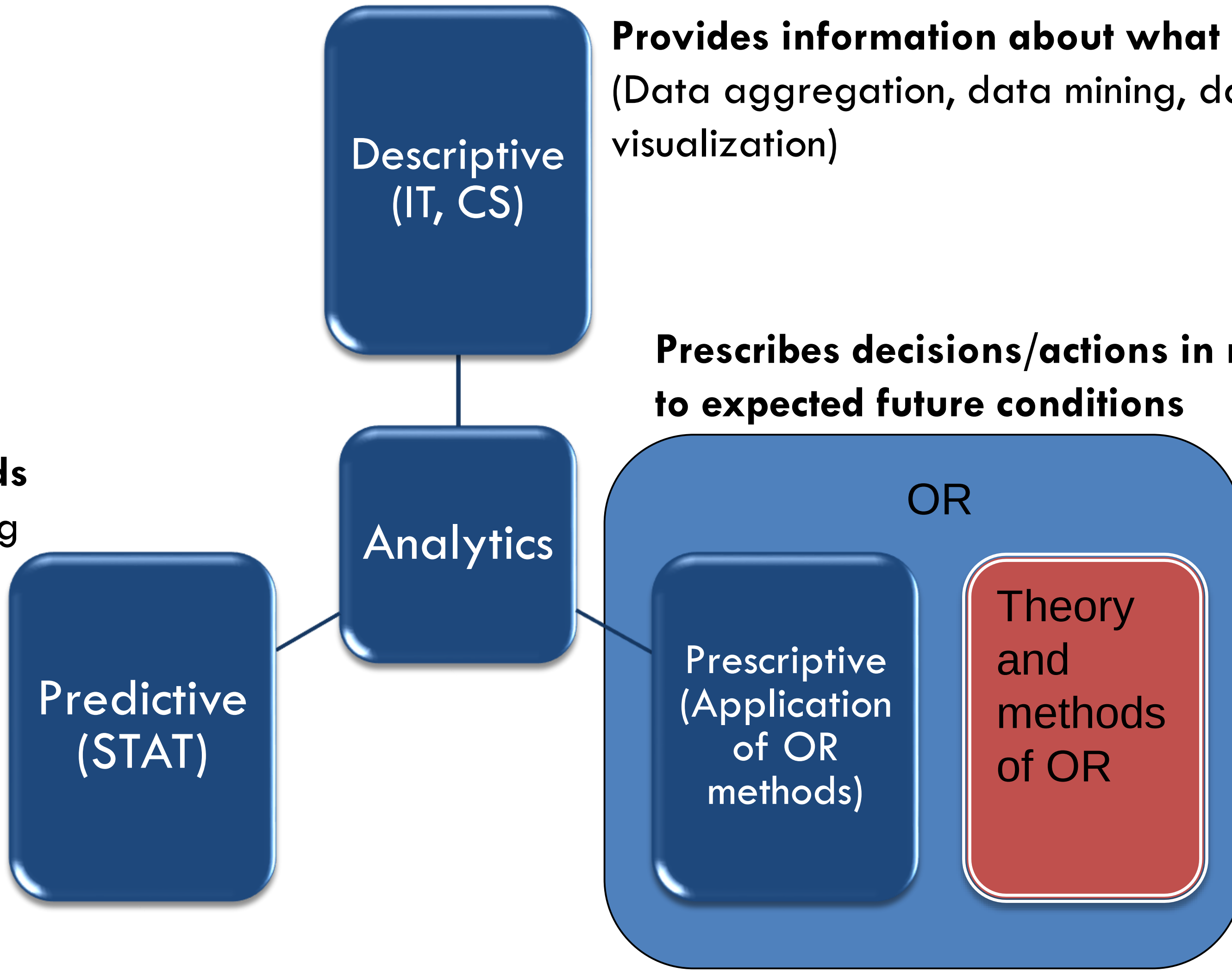
Ankit Shah
Ph.D. Candidate

Analytics and Operations Research (OR)

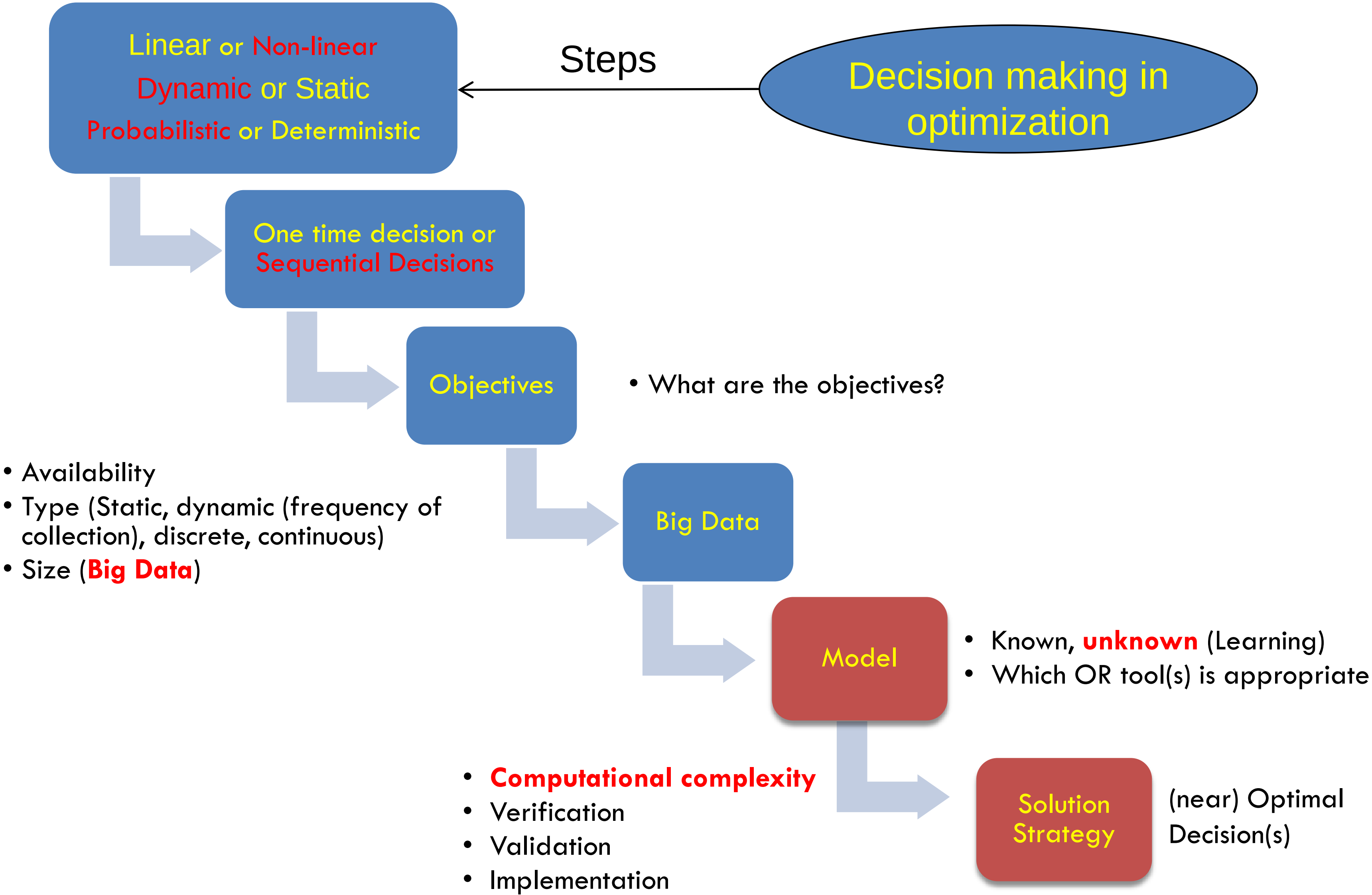
Provides information about what happened
(Data aggregation, data mining, data visualization)

Prescribes decisions/actions in response to expected future conditions

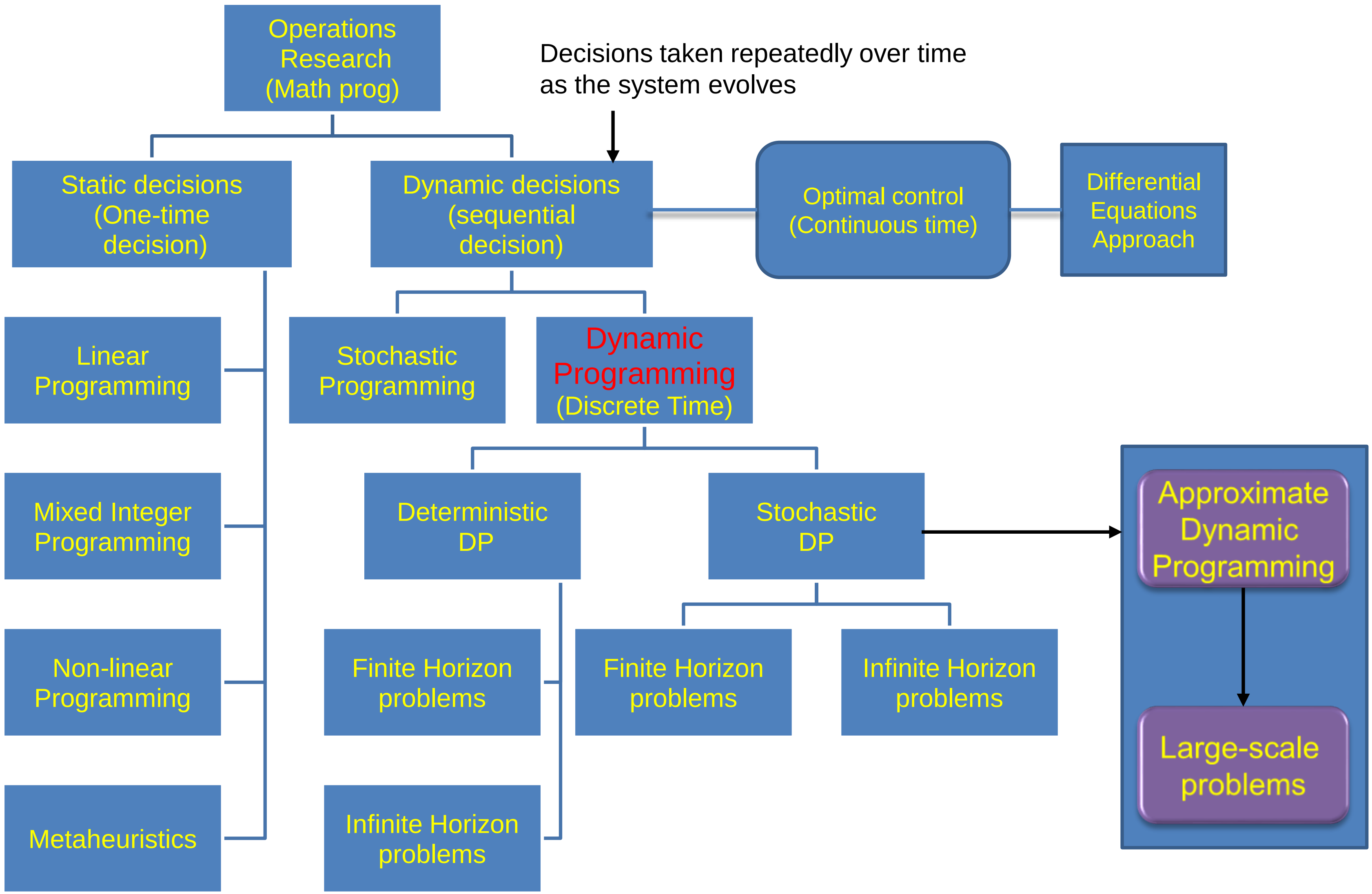
Identifies future possibilities/trends
(Statistical learning algorithms)



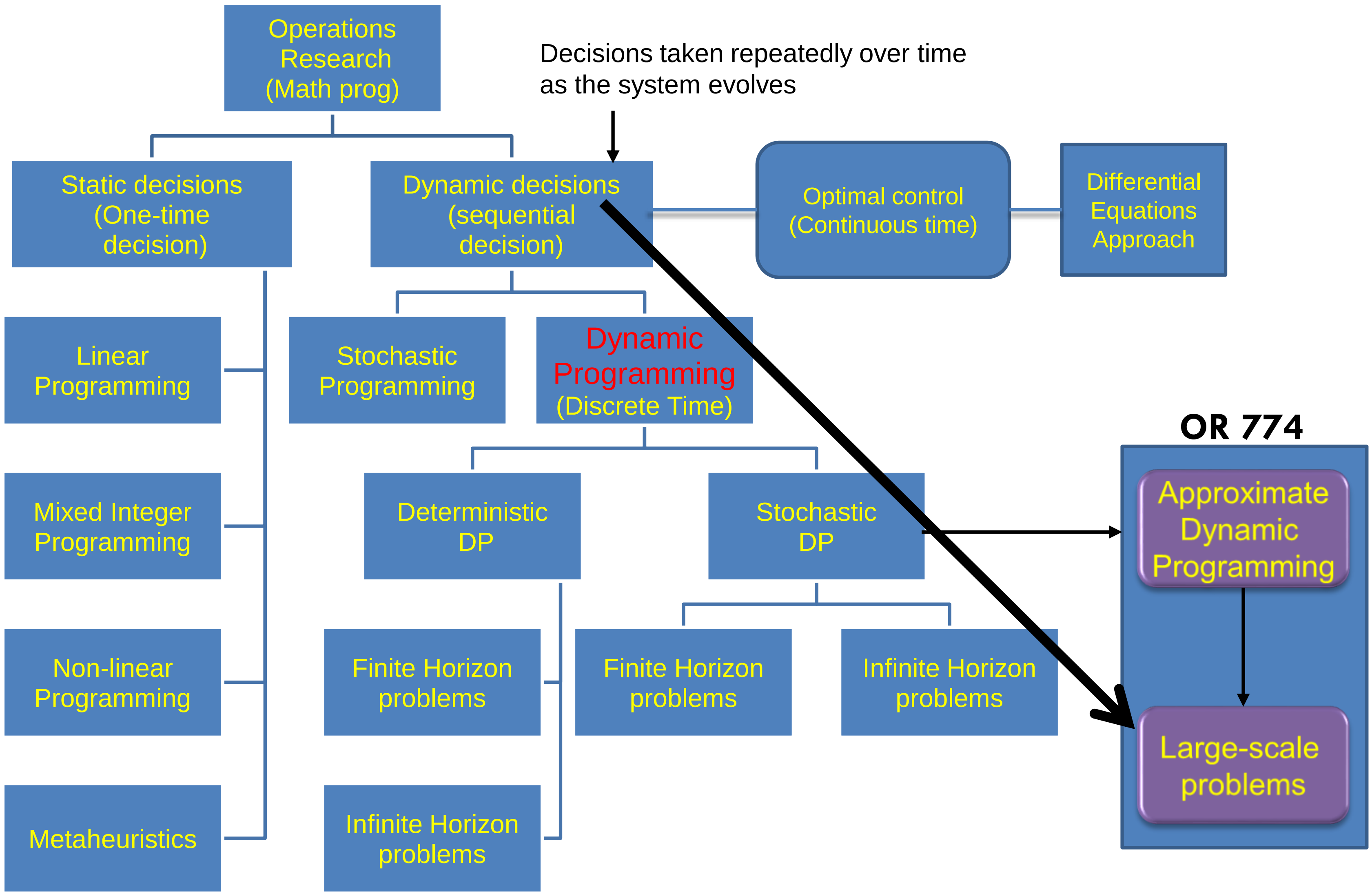
Decision Making in Optimization



The Big Picture



The Big Picture



Static Decision Making

- Static (ignore time)
  - One-time Investment
  - Assignment
    - People to jobs
    - Jobs to machines (maximize throughput, minimize tardiness)
    - Classrooms to courses
  - Traveling Salesman (leave home city, travel to different cities and return back to home city in the shortest distance without revisiting a city)
  - Set Covering (Installation of fire stations)

Dynamic Decision Making

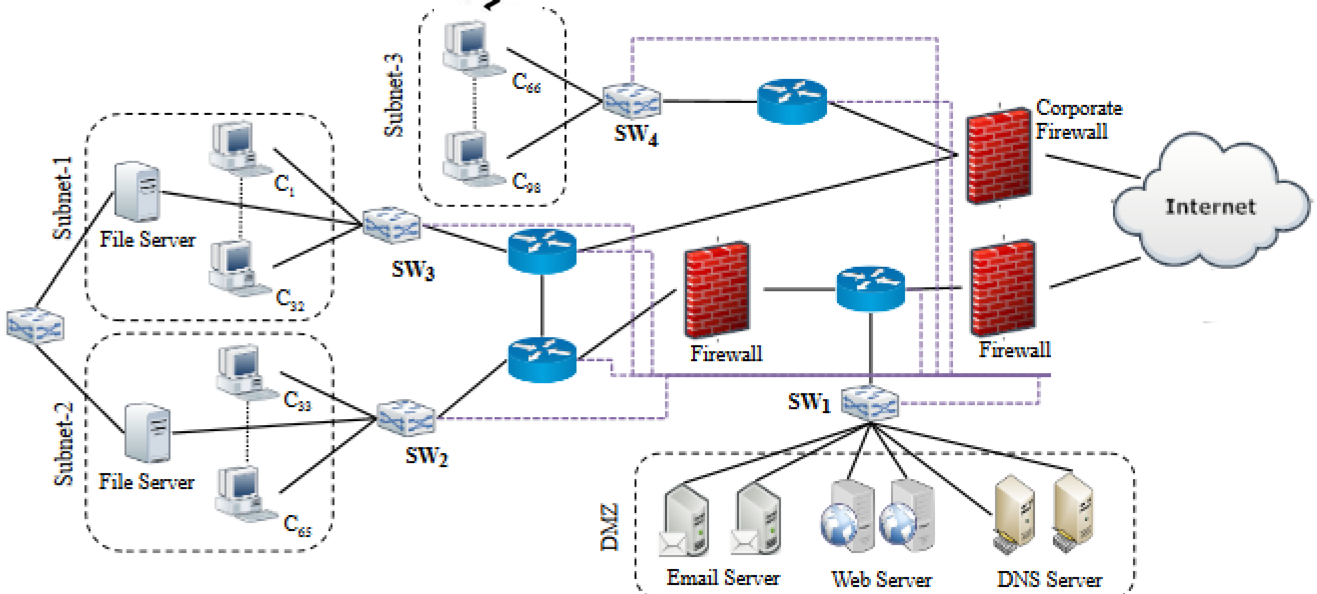
- Dynamic (several decisions over time)
  - Portfolio management (daily trading)
  - Inventory control (hourly, daily, weekly, …)
  - Dynamic Assignment
    - Running a shuttle company (by the minute)
  - Airline seat pricing (by the hour)
  - Air traffic control (by the minute)
  - Maneuvering a combat aircraft or a helicopter or a missile (decisions every millisecond)

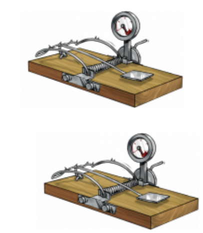
An Example

Stochastic Dynamic Programming Framework for a Network Security Problem

Organization’s Network

Malicious Activity



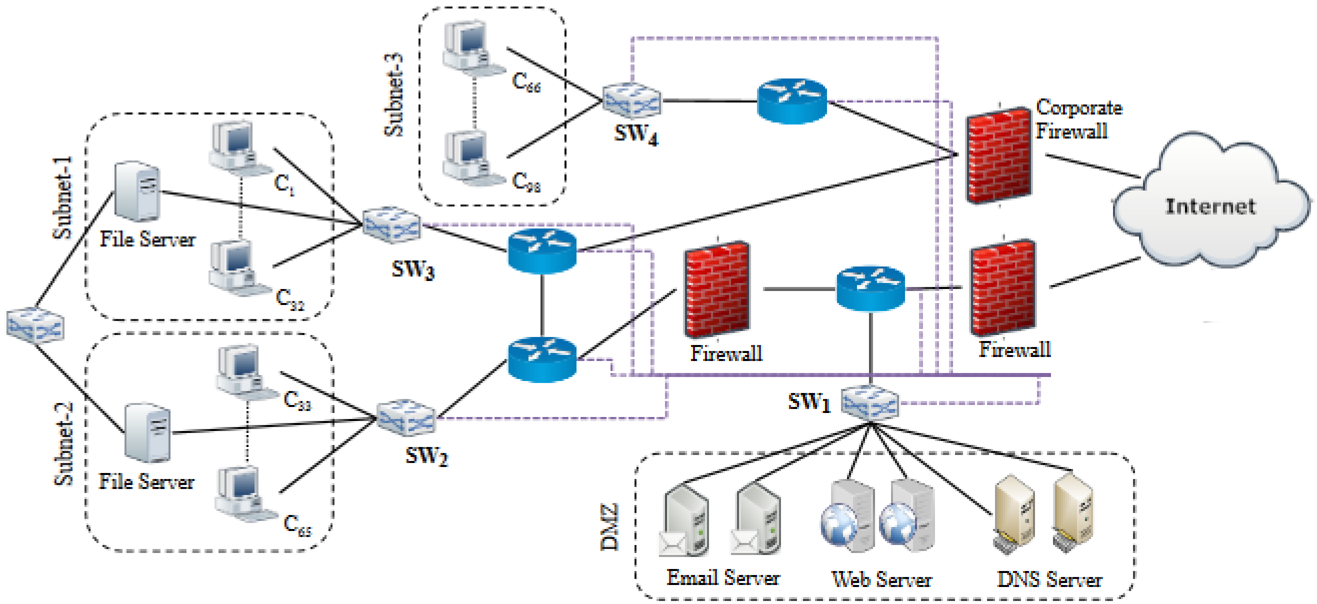


[Venkatesan et al. 2017]

Monitors/Honeypots

Organization’s Network

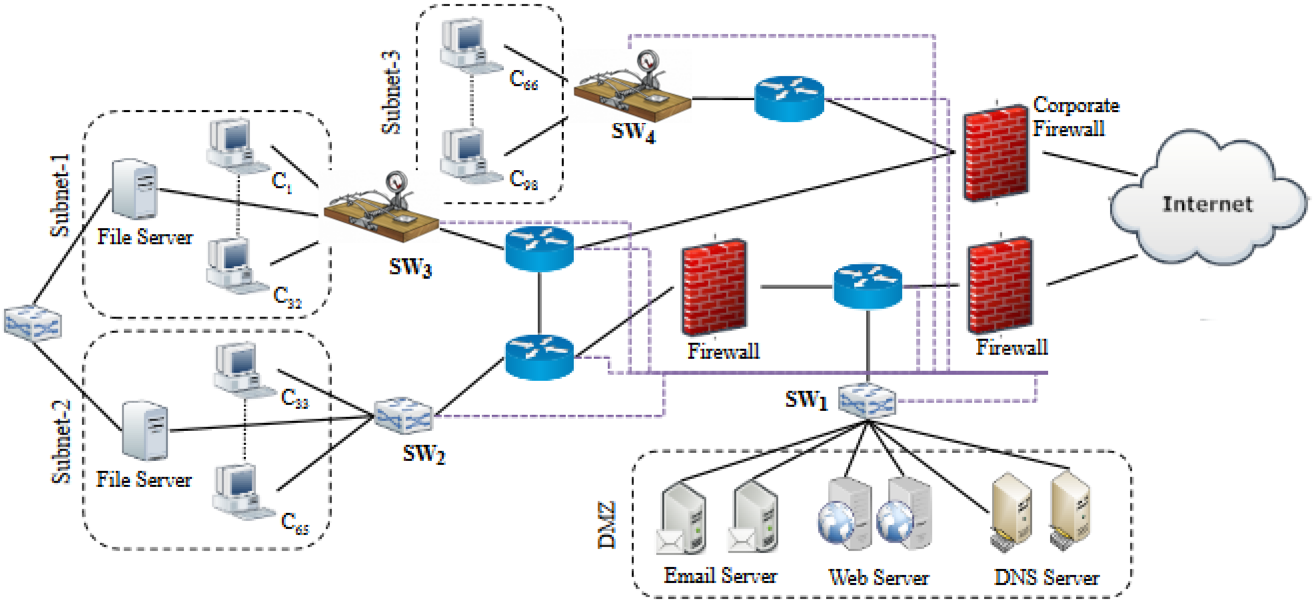
Objective: Decide the **placement of monitors** at each epoch (time index “t”) such that **maximum number of malicious activities** are identified and mitigated (**infinite horizon**).



[Venkatesan et al. 2017]

Organization's Network

Objective: Decide the placement of monitors at each epoch (time index "t") such that maximum number of malicious activities are identified and mitigated (infinite horizon).

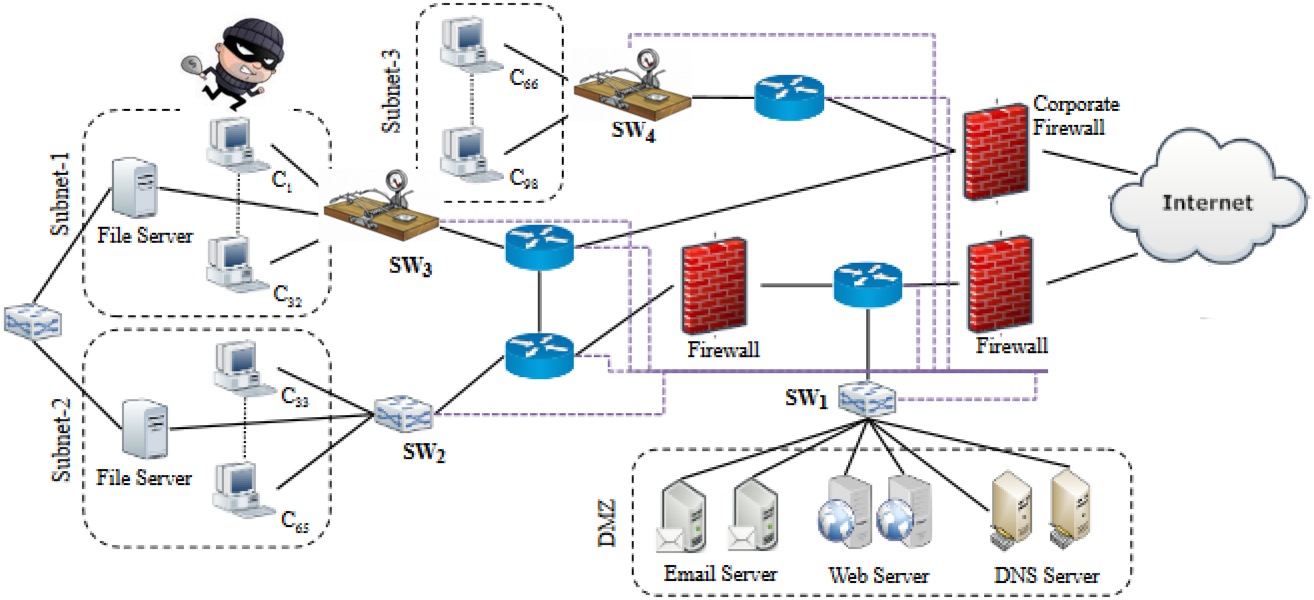


time = t

[Venkatesan et al. 2017]

Organization’s Network

Objective: Decide the placement of monitors at each epoch (time index “t”) such that maximum number of malicious activities are identified and mitigated (infinite horizon).

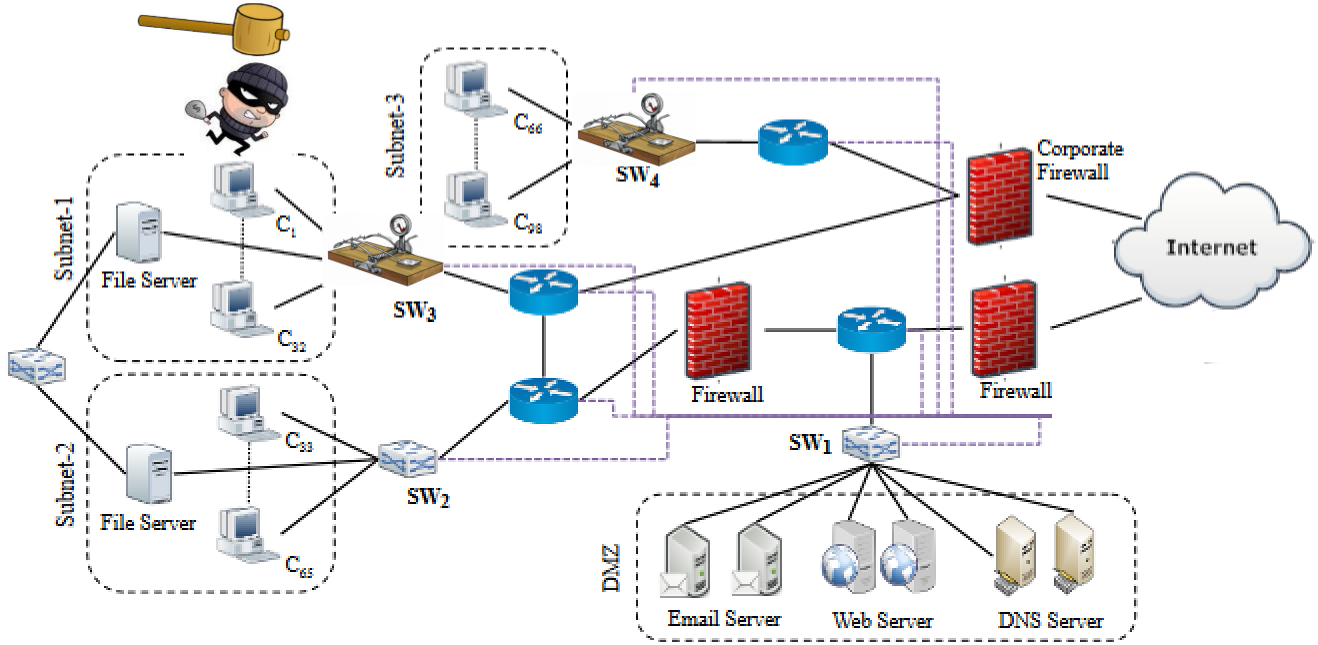


Between t to t+1

[Venkatesan et al. 2017]

Organization's Network

Objective: Decide the placement of monitors at each epoch (time index "t") such that maximum number of malicious activities are identified and mitigated (infinite horizon).

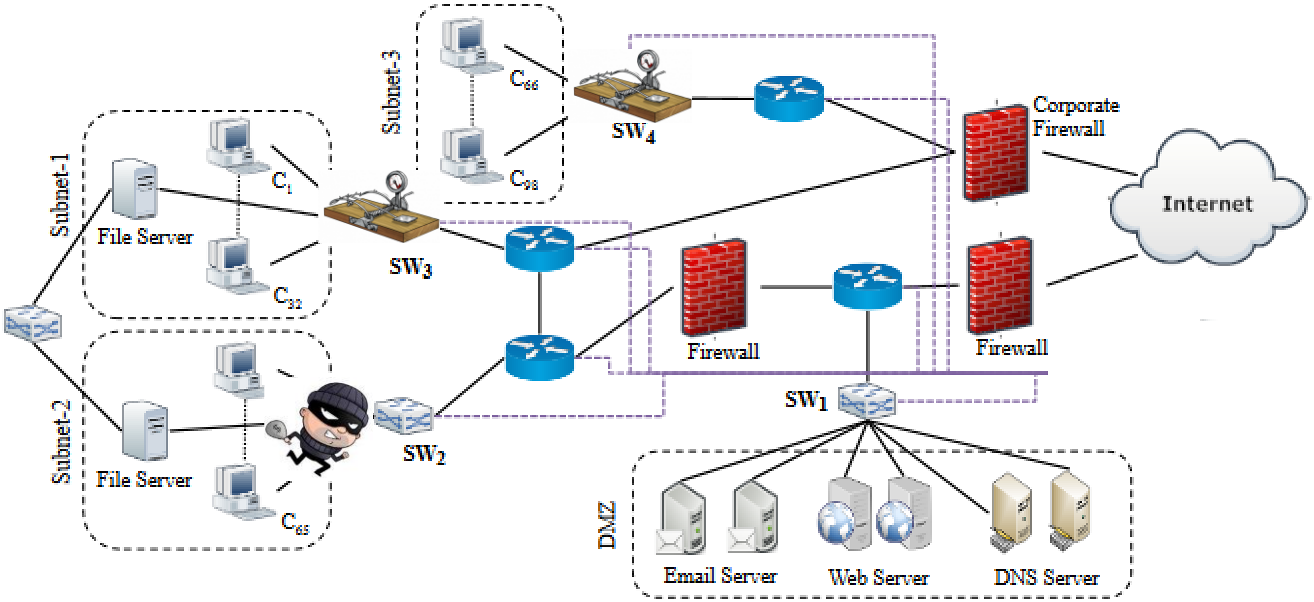


Between t to t+1

[Venkatesan et al. 2017]

Organization's Network

Objective: Decide the placement of monitors at each epoch (time index "t") such that maximum number of malicious activities are identified and mitigated (infinite horizon).

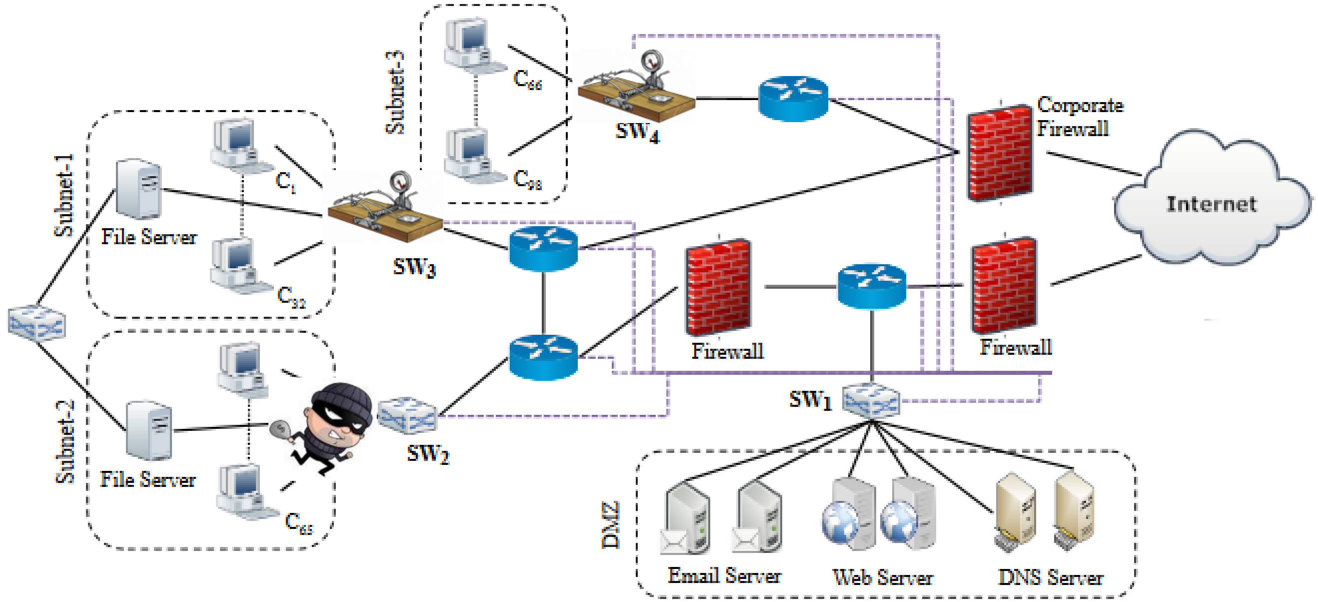


Between t to t+1

[Venkatesan et al. 2017]

Organization's Network

Objective: Decide the placement of monitors at each epoch (time index "t") such that maximum number of malicious activities are identified and mitigated (infinite horizon).



Between t to t+1

[Venkatesan et al. 2017]

Sequential Decision Making under **Uncertainty** with an **Unknown Model** → **Solved using Stochastic Dynamic Programming (Learn through Simulation-based Optimization)**

Today's Talk

- Modeling and Solution Strategies for Static and Dynamic Decision Making
  - Linear Programming example
  - Integer Programming example
- What to do if the model is too hard to obtain or its simply not available and there is high computational complexity
  - Metaheuristics (directly search the solution space)
  - Simulation-based Optimization
- Dynamic Programming example
- Computational aspects

Linear Programming

- 100 workers
- 80 acres of land
- 1 acre of land produces 1 ton of wheat/corn
- 2 workers are needed for every ton of either crop
- Storage permits only a max production of 40 tons of wheat
- Selling price of wheat = $3/ton
- Selling price of corn = $2/ton

Mathematical Model

$$Max\ Z = 3x_1 + 2x_2$$

Subject to

$$\begin{aligned} 2x_1 + 2x_2 &\leq 100 \\ x_1 + x_2 &\leq 80 \\ x_1 &\leq 40 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

- x1 = quantity of wheat to grow in # of tons
- x2 = quantity of corn to grow in # of tons

- How many tons of wheat and corn to produce to maximize revenue?
  - Solution: Simplex Algorithm, solved using solvers, CPLEX, Gurobi.

Assumptions in Linear Programming

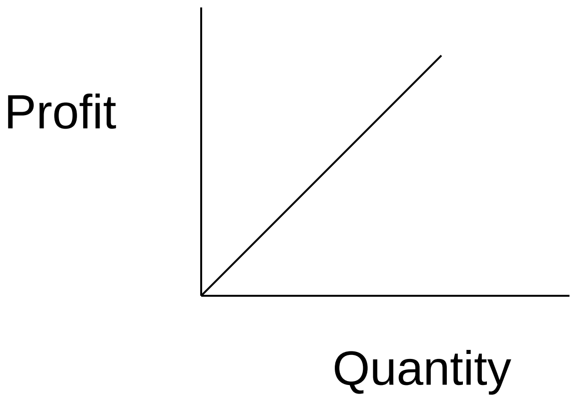
- 1. Proportionality: This is guaranteed if the objective and constraints are linear
- 2. Additive: Independent decision variables
- 3. Divisibility: Fractions allowed
- 4. Certainty: Coefficients in the objective function and constraints must be fixed

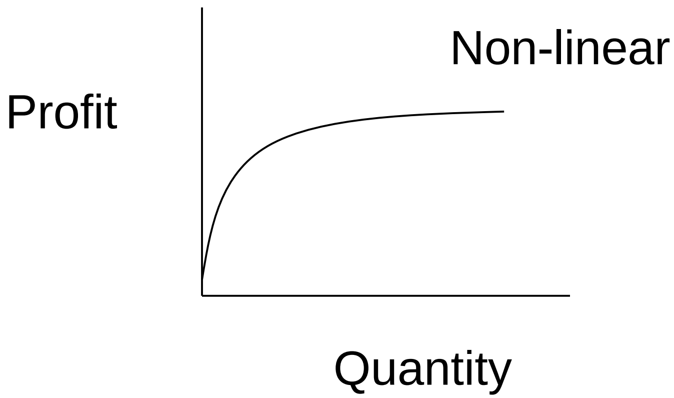
What if you had many decision variables

- Big Data
- Computational burden
  - Today's solvers can handle large problems
- Linear Programming is easy to implement
- However, solutions can be far from optimal if applied to problems under uncertainty in a non-linear environment
  - Use only when appropriate
- Is the real-world linear, fixed, deterministic?

Relax Assumption 1

- 1. Proportionality: if not true
- Max $Z = 3x_1^2 + 2x_2^2$
- Need Non-linear Programming (far more difficult than Linear Programming)
- Solution strategies are very different
  - Method of steepest ascent, Lagrangian Multipliers, Kuhn-Tucker methods
- OR 644 - A separate course taught by Dr. Sofer





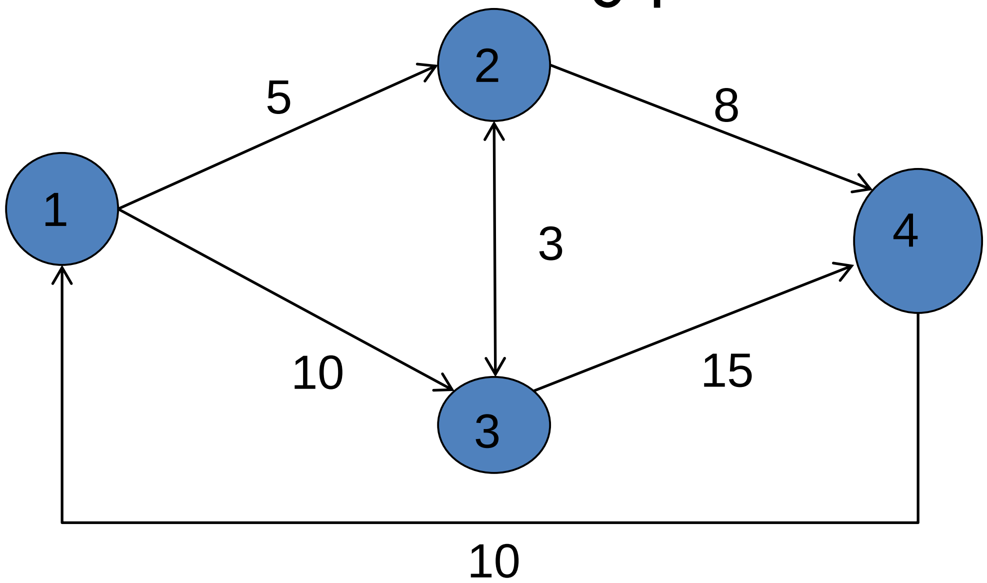
Relax Assumption 3

- 3. Divisibility: If fractions are not allowed
- Yes or no decisions (0,1) binary variables
- Assignment problems
- Need Integer Programming
- OR 642 - A separate course taught by Dr. Hoffman
- These problems are more difficult to solve than Linear Programming

Examples

Example

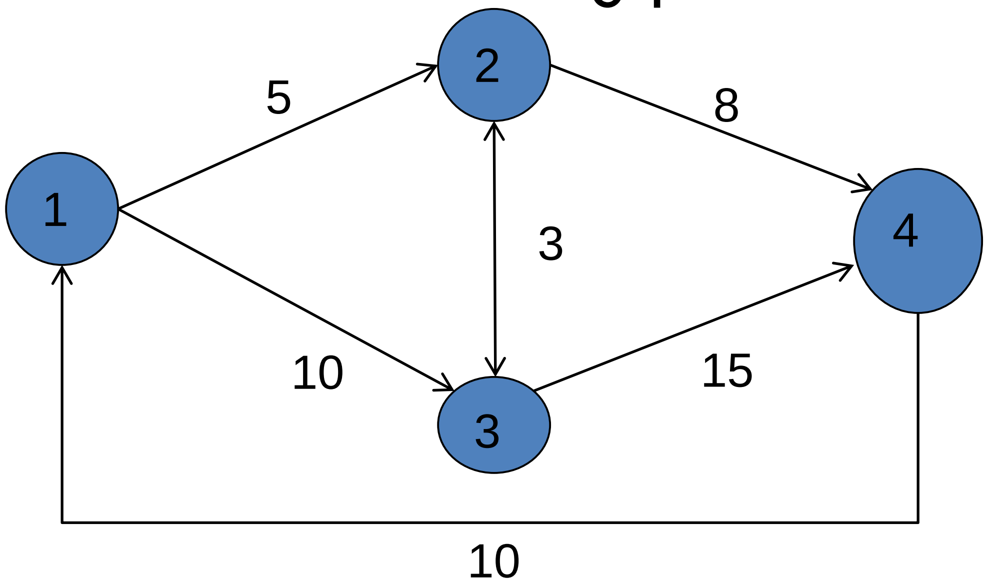
- Consider the following problem:



- Find the shortest route starting at node 1 such that:
  - the selected route passes each node exactly once,
  - and comes back to node 1.

Example

- Consider the following problem:

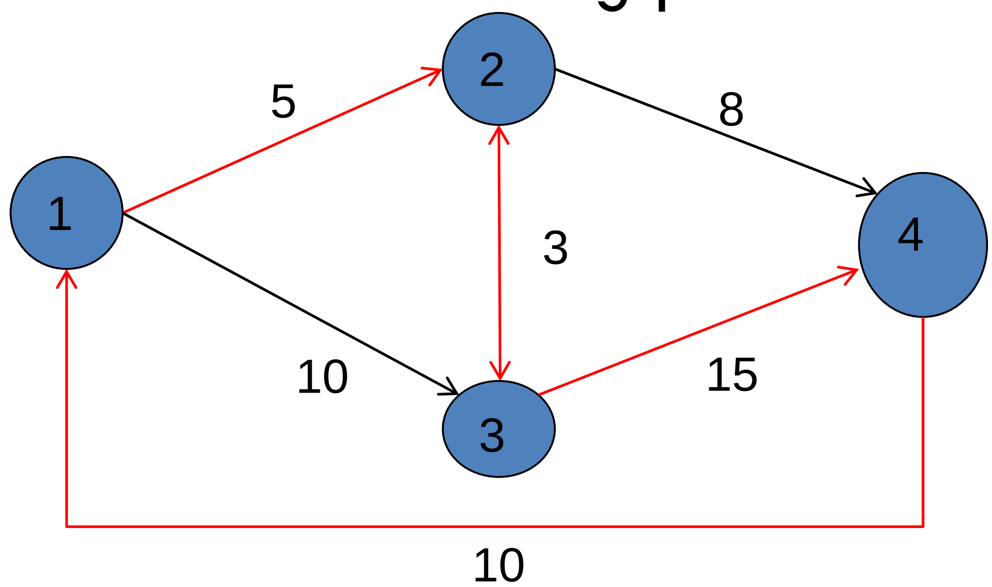


Also known as a Traveling Salesman Problem (TSP)
(Finding the shortest path covering all the cities)

- Find the shortest route starting at node 1 such that:
  - the selected route passes each node exactly once,
  - and comes back to node 1.

Example

- Consider the following problem:



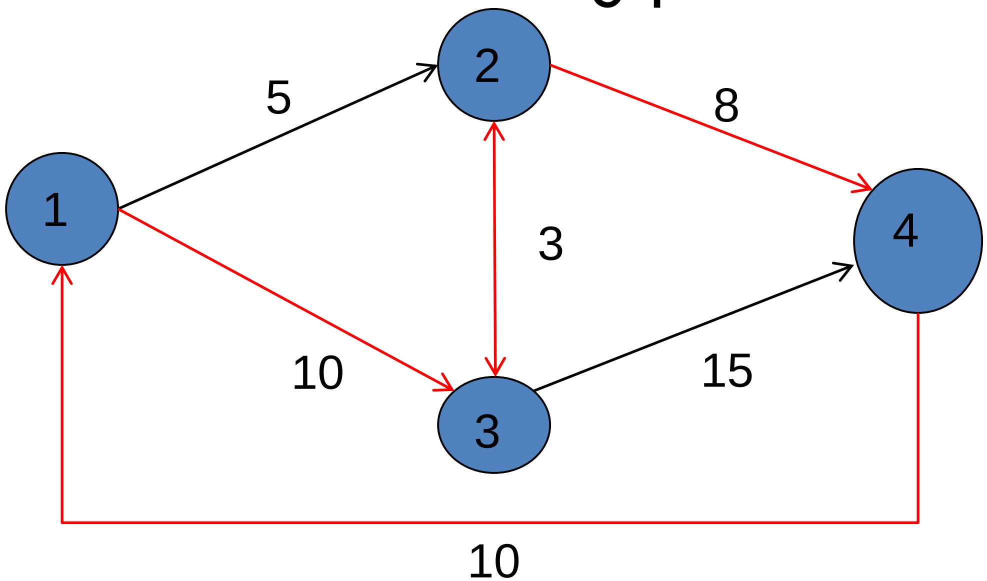
Also known as a Traveling Salesman Problem (TSP)

(Finding the shortest path covering all the cities)

- Myopic Route:1-2-3-4-1 = 5+3+15+10 = 33

Example

- Consider the following problem:



Also known as a Traveling Salesman Problem (TSP)

(Finding the shortest path covering all the cities)

- Myopic Route:1-2-3-4-1 = 5+3+15+10 = 33
- Optimal Route: 1-3-2-4-1 = 10+3+8+10 = 31

Computational Complexity

- Now, imagine solving a TSP for 20 cities.
  - An exhaustive enumeration: $20! = 2*10^{18}$ solutions.
  - If a computer can evaluate 100 million solutions/second, it will take 771 years.

Example

- Consider the following problem:

Item	benefit	weight
1	60	10
2	100	20
3	120	30

- Maximize your benefit.
- You can pick an item or a fraction of an item.
- Total weight must not exceed 50.

Example

- Solution Method:

Item	benefit	weight	benefit/weight
1	60	10	6
2	100	20	5
3	120	30	4

- Greedy Approach (Pick in an order: High to low)
  - Pick item 1 benefit = 60 weight = 10
  - Pick item 2 benefit = 100 weight = 20
  - Pick 2/3 of item 3 benefit = 80 weight = 20

Total benefit = 240 weight = 50

Example

- Consider the same problem:

Item	benefit	weight
1	60	10
2	100	20
3	120	30

Also known as a 0-1 Knapsack Problem
(for ex. selecting items for carry-on bag with a weight restriction.)

- Maximize your benefit.
- **<u>You can pick an item (fraction of an item is not allowed)</u>.**
- Total weight must not exceed 50.

Knapsack Problem

- Solution Method:

Item	benefit	weight	benefit/weight
1	60	10	6
2	100	20	5
3	120	30	4

- Greedy Approach (Pick in an order: High to low)
  - Pick item 1 benefit = 60 weight = 10
  - Pick item 2 benefit = 100 weight = 20
  - ~~Pick 2/3 of item 3 benefit = 80 weight = 20~~

Total benefit = 160 weight = 30

Knapsack Problem

- Solution Method:

Item	benefit	weight	benefit/weight
1	60	10	6
2	100	20	5
3	120	30	4

- **Optimal Selection**:
  - Pick item 3 benefit = 120 weight = 30
  - Pick item 2 benefit = 100 weight = 20

Total benefit = 220 weight = 50

Knapsack Problem

- Solution Method:

Item	benefit	weight	benefit/weight
1	60	10	6
2	100	20	5
3	120	30	4

- **Integer Programming Formulation**:

  Max 60 x1 + 100 x2 +120 x3

  Subject to

  10 x1 + 20 x2 + 30 x3 $\leq$ 50

  where x1,x2, and x3 are binary variables (either 0 or 1)

Computational Complexity

- Now, try packing a UPS/FEDEX truck or aircraft with both weight & volume constraints and maximize the benefit.
  - Although computers can help to solve, the solution will be computationally very expensive for large real-world problems.
  - In many cases, we strive of near-optimal (good enough) solutions.

Near-Optimal Solution Techniques

Metaheuristics (OR 670)

- Several techniques (Genetic algorithm, simulated annealing, tabu search, …)
- Search the solution space
- There are no models like LP, IP, NLP
- Start your search by defining one or many feasible solutions
- Improve your objective of the search by tweaking your solutions systematically
- Stop search when you have had enough of it (computing time reaches your tolerance)
- Be happy with the solution that you have at that point
- You may have gotten the optimal solution but you will never know that it is indeed optimal
- Metaheuristics is <u>not suitable for sequential decision making under probabilistic conditions (uncertainty)</u>

Sequential Decision Making under Uncertainty

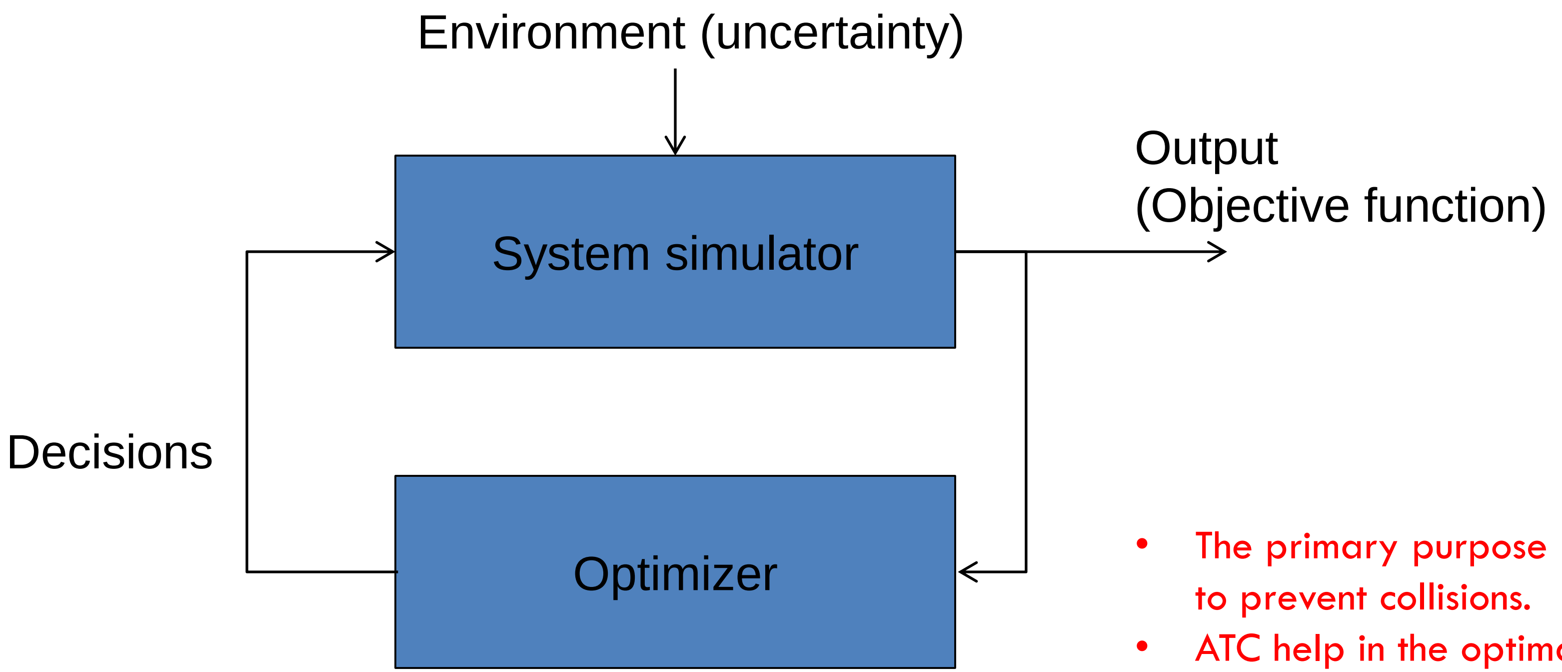
- Let us introduce dynamic decisions **over time** and **uncertainty** (stochastic behavior)

on top of

- Big data
- Complex non-linear system
- Computational difficulty (state space and dimensionality)
- Time between decisions too short

Simulation-based Optimization

- Model-free Approach



- The primary purpose is to prevent collisions.
- ATC help in the optimal movement of air traffic.

- Simple example: Car on cruise control
  - A mathematical model that relates all car parameters and the environment parameters may not exist
- A more difficult to solve and complex example: Air traffic control
  - (Optimizer is an Artificial Intelligence (Learning) Agent)

How does an AI agent learn?

- In a discrete setting you need Dynamic Programming (OR674 and OR 774) – term common among advanced OR
- In a continuous time setting it is called optimal control (Differential equations are used) – term common among Electrical Engineers
- Mathematically the above methods are IDENTICAL
- Computer Science folks call it machine learning, AI, or Reinforcement Learning and use it mainly for computer games

AlphaGo Zero (2017):

Acquired 3000 years of human knowledge in 40 days from scratch, simply by playing millions of games against itself.

Learned the **best moves over time** and **developed new strategies.**

Different Lines of Investigation

- Operations Research – Markov Decision Processes
  - Bellman, 1957
  - Powell, 2007
- Control Theory – Heuristic Dynamic Programming / Neuro Dynamic Programming
  - Problems in physical processes with continuous states and actions
  - Werbos, 1974
  - Bertsekas and Tsitsiklis, 1996
- Computer Science - Reinforcement Learning
  - Samuel, 1959
  - Sutton and Barto, 1981

Dynamic Programming

- What is Dynamic Programming (DP)?
  - An optimization method that finds the shortest path (ex: minimize cost) or the longest path (ex: maximize reward) in decision making problems that are solved sequentially over time.

Dynamic Programming

- What is Dynamic Programming (DP)?
  - An optimization method that finds the shortest path (ex: minimize cost) or the longest path (ex: maximize reward) in decision making problems that are solved sequentially over time.

Deterministic Dynamic Programming:

Week 1	Course introduction, Finite Decision Trees
Week 2	Dynamic Programming Networks and the Principle of Optimality
Week 3	Formulating dynamic programming recursions, Shortest Path Algorithms, Critical Path Method, Resource Allocation (including Investments)
Week 4	Knapsack Problems, Production Control, Capacity Expansion, and Equipment Replacement
Week 5	Infinite Horizon Optimization including Equipment Replacement over an Unbounded Horizon
Week 6	Infinite Decision Trees and Dynamic Programming Networks

Dynamic Programming

- What is Dynamic Programming (DP)?
  - An optimization method that finds the shortest path (ex: <u>minimize</u> cost) or the longest path (ex: <u>maximize</u> reward) in decision making problems that are solved <u>sequentially over time</u>.

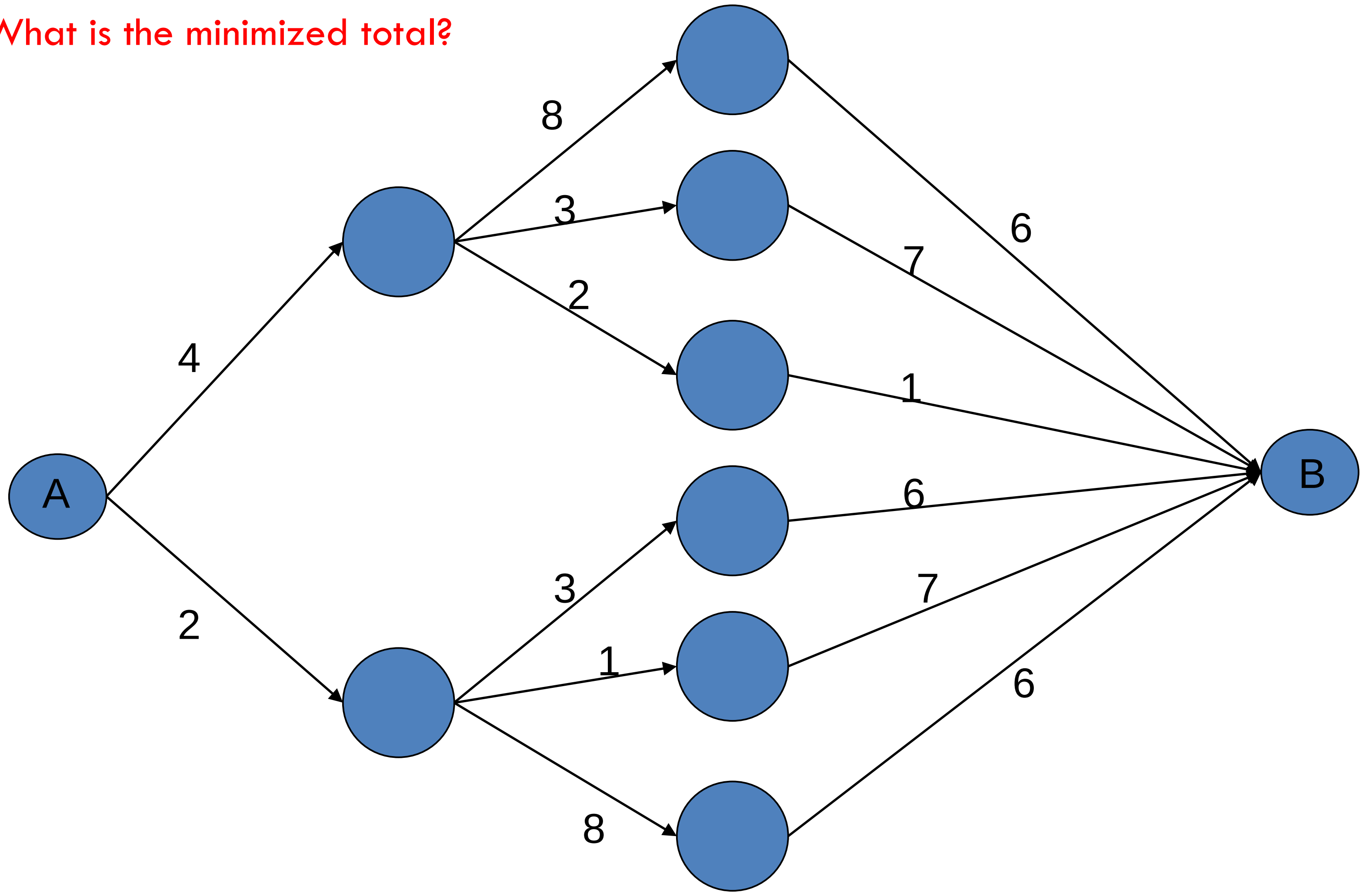
Stochastic Dynamic Programming

Week 8	Stochastic Shortest Path Problems with examples in Inventory Control
Week 9	Markov Decision Processes, value and policy iteration for average cost criteria
Week 10	Markov Decision Processes, value and policy iteration for discounted cost criteria
Week 11	Markov Decision Processes, value and policy iteration for discounted cost criteria
Week 12	MDP with examples in Equipment Replacement and inventory problems
Week 13	Semi-Markov Decision Process
Week 14	Semi-Markov Decision Process

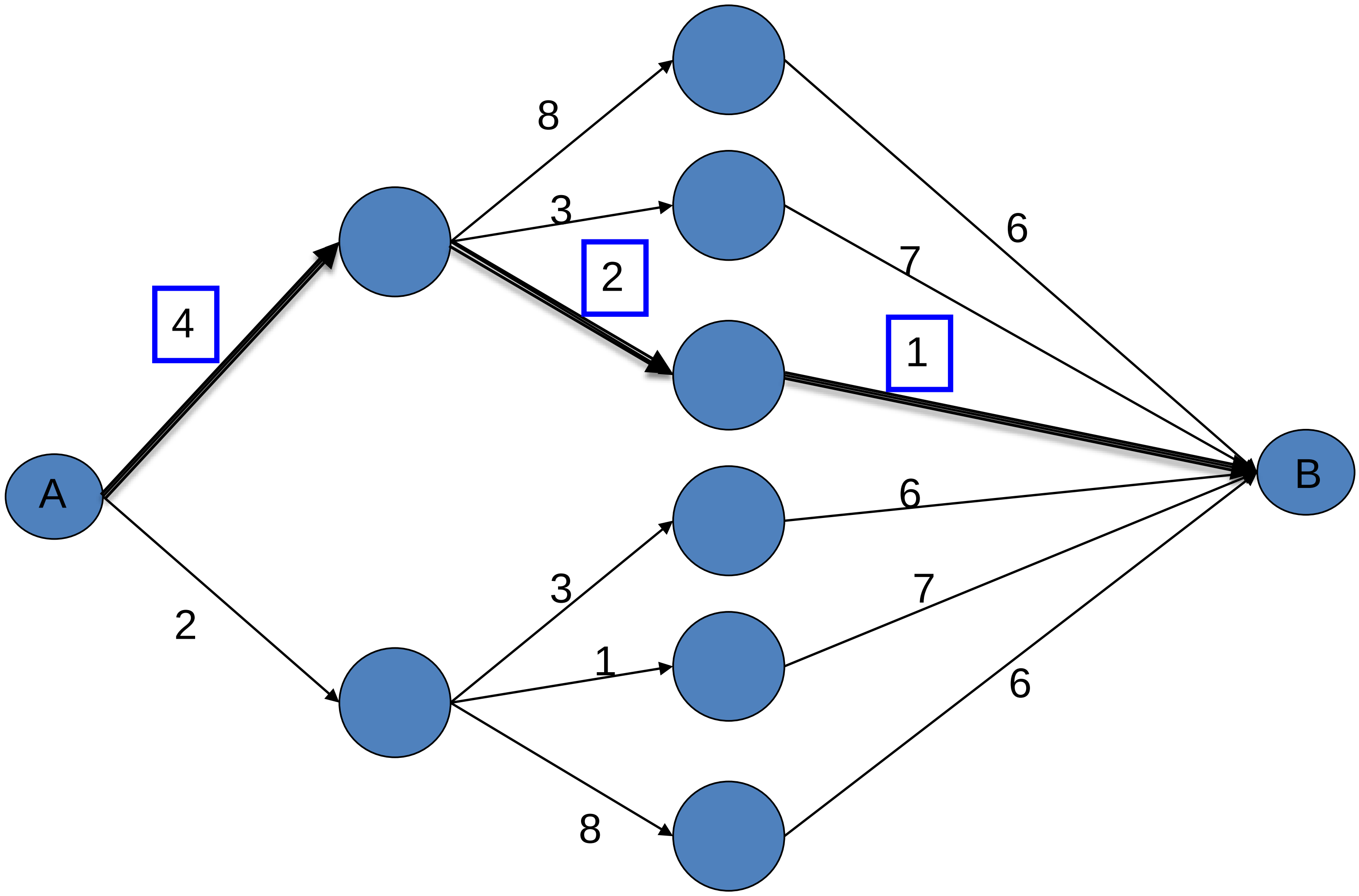
An Example

Find Shortest Path from A to B

What is the minimized total?



Find Shortest Path from A to B



Questions

- How many of you evaluated all possible paths to arrive at the answer?
- How many of you started by looking at the smallest number from A (in this case it is 2) and went on to the next node to find the next smallest number 1 to add and then added 7 to get an answer of 10

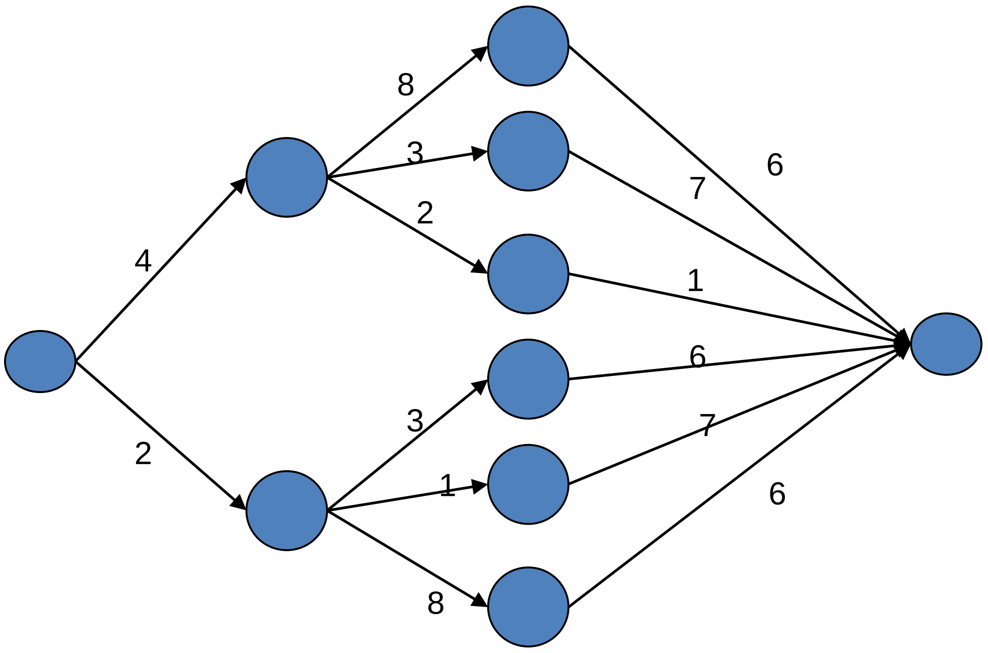
- If you did all possible paths then you performed an explicit enumeration of all possible paths (you will need 771 years or more to solve 20 city TSP)

or

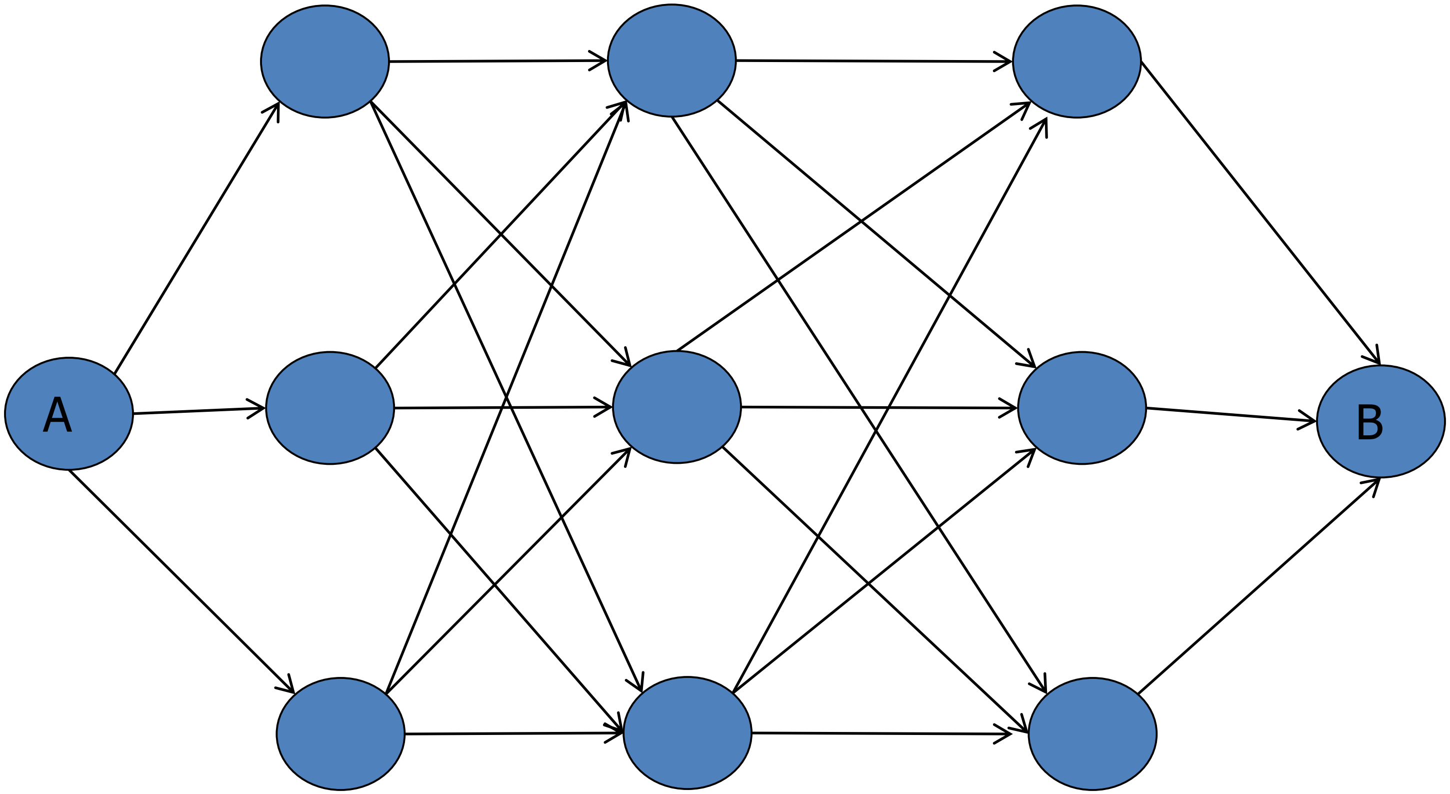
- you tried to follow a myopic (short-sight) policy, which did not give the correct answer

Computational Perspective

- For explicit enumeration, to find the shortest path
  - There were 18 additions
  - And 5 comparisons (between 6 paths)

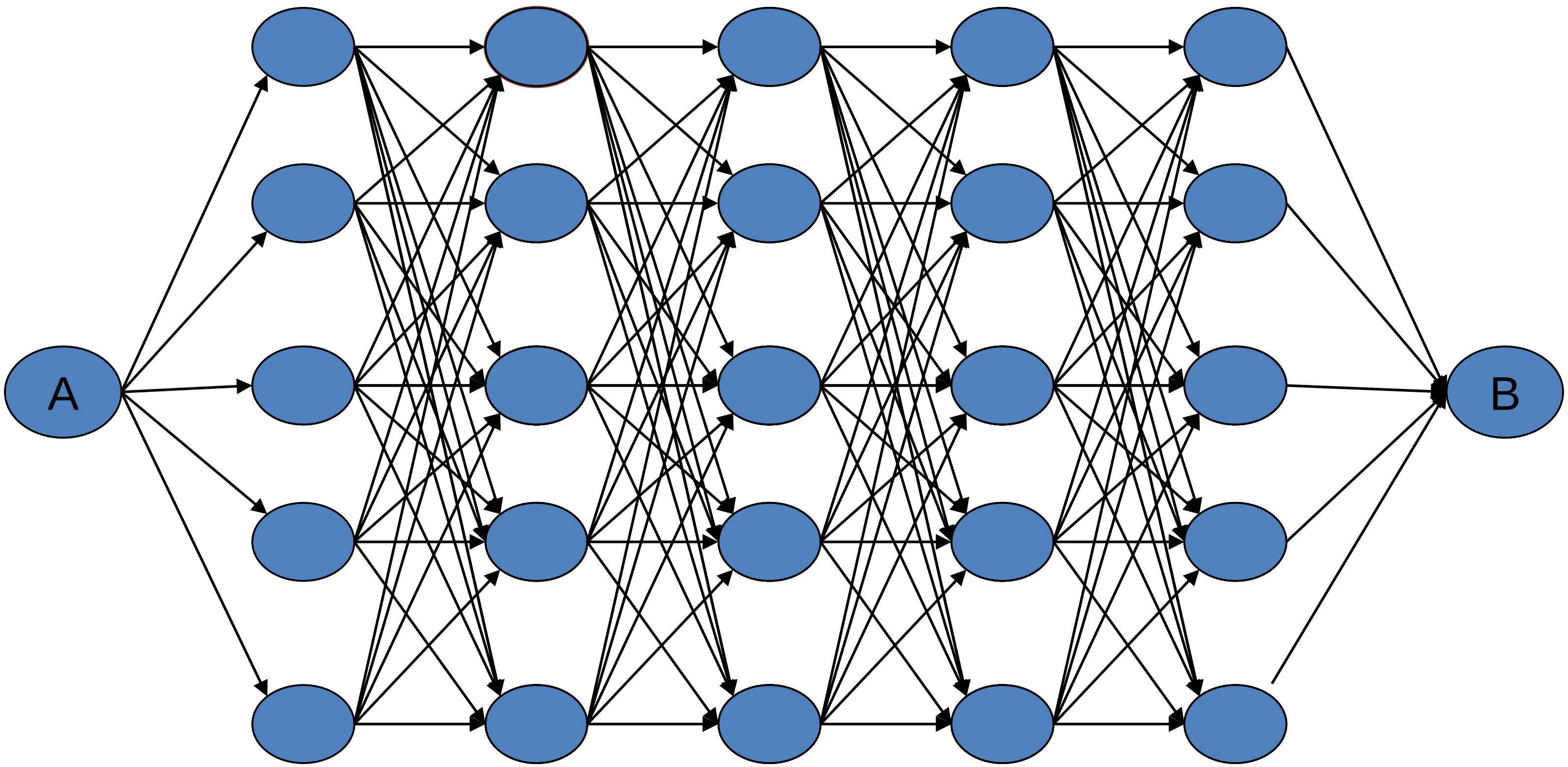


Another Example



Explicit enumeration
27 paths
27*3= 81 additions
26 comparisons

Another Example



Explicit enumeration

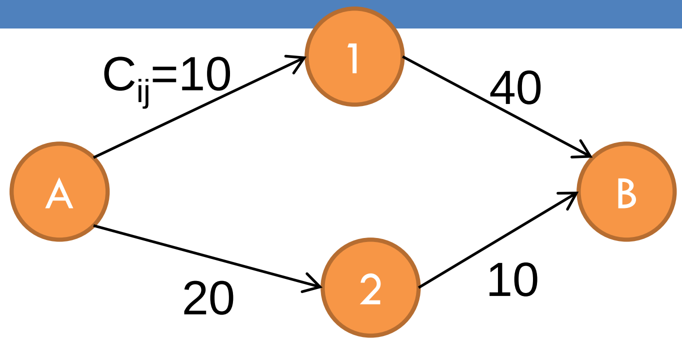
5^5 paths*5 additions per path=15625 additions

$5^5 - 1$ comparisons = 3124

Dynamic Programming Approach

Myopic vs Dynamic Programming

C_{ij}= cost on the arc

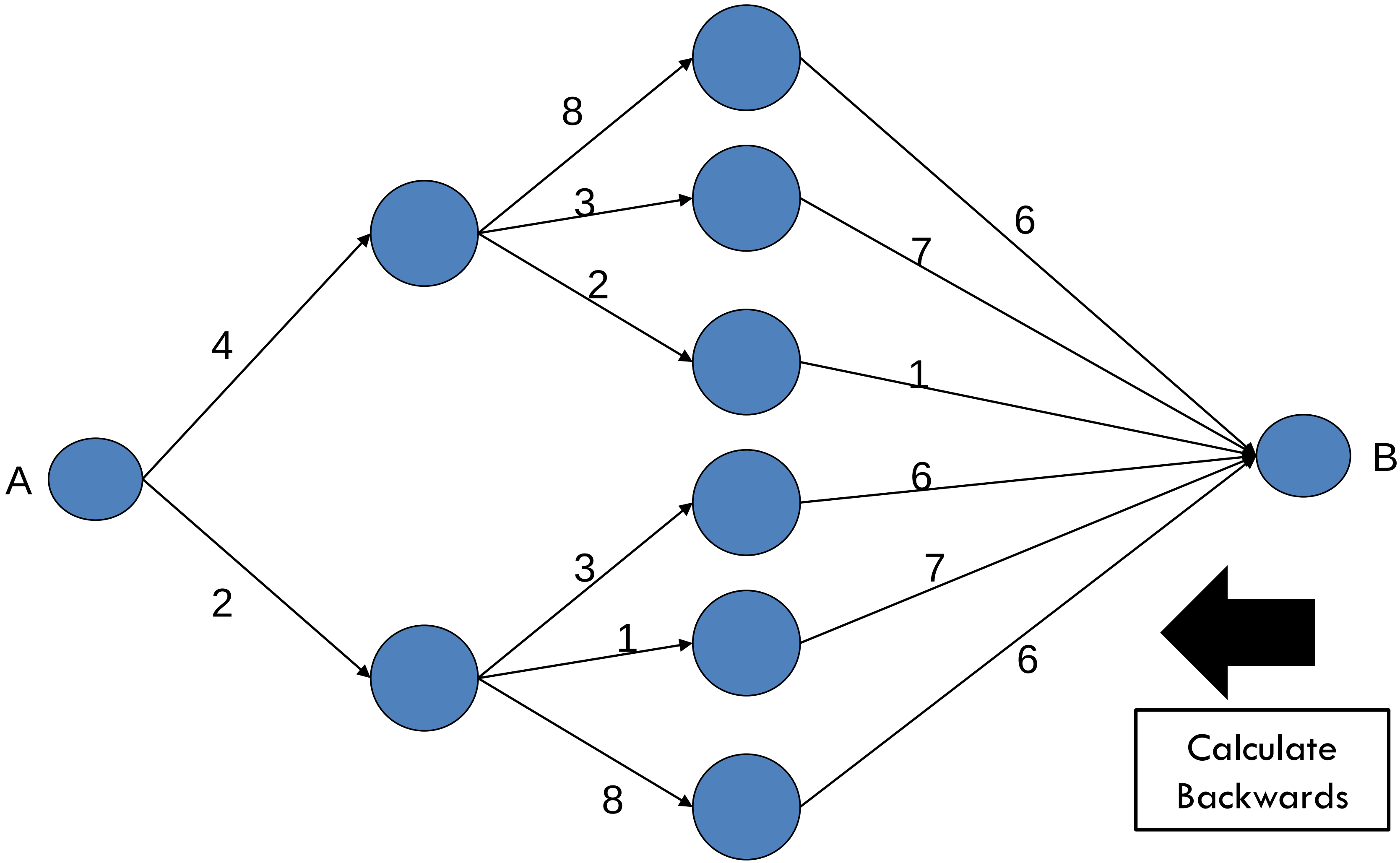


Myopic policy: $V(A) = \min(C_{ij})$
$= \min \text{ of } (10 \text{ or } 20)$
leads to solution of 50 from A to 1 to B

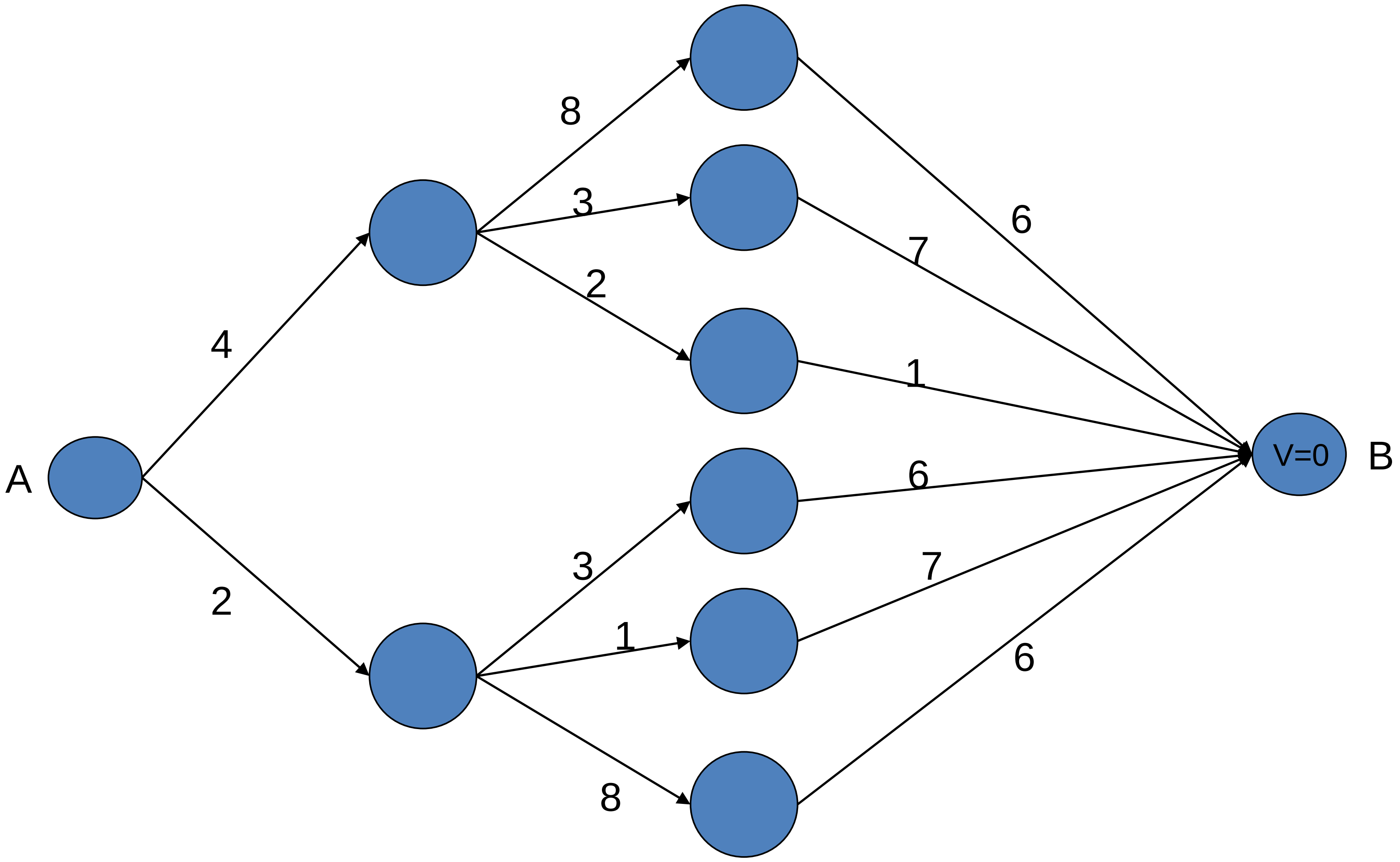
DP policy: $V(A) = \min(C_{ij} + V(\text{next node}))$
$= \min(10 + 40, 20+10) = 30$
leads to solution of 30 from A to 2 to B

Key is to find the values of node 1 and 2
How? By learning via simulation-based optimization

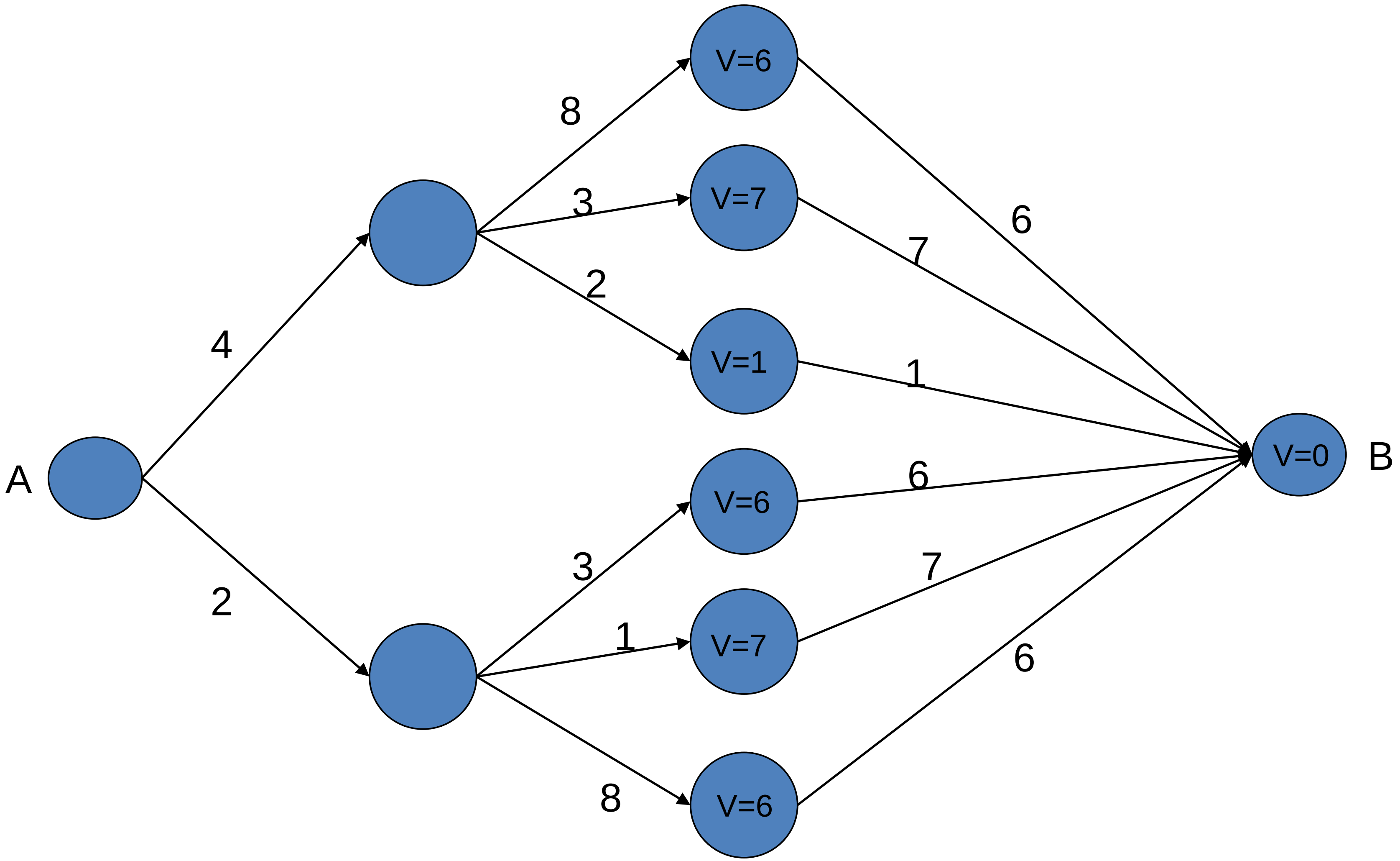
Find Shortest Path from A to B (using DP)



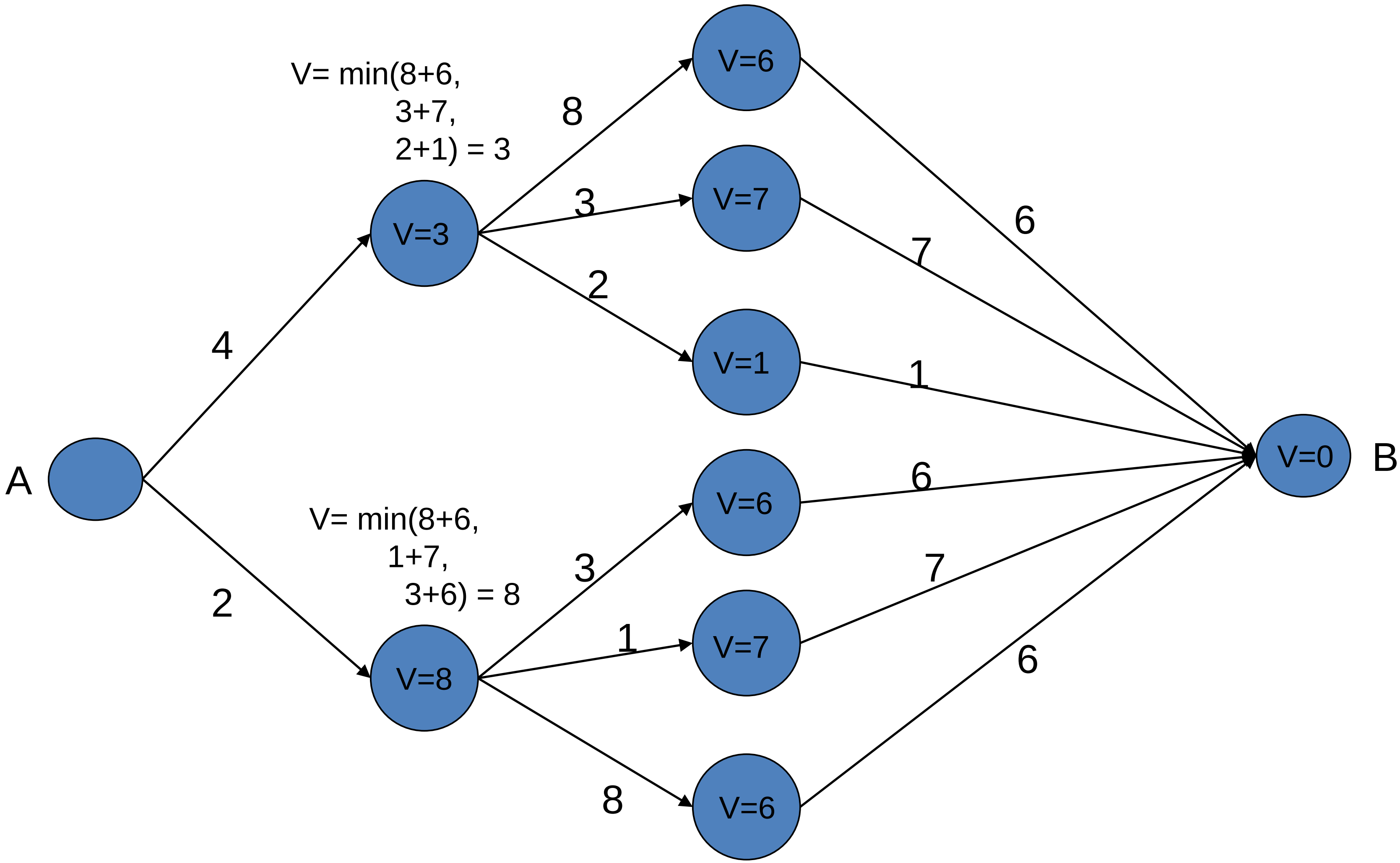
Find Shortest Path from A to B (using DP)



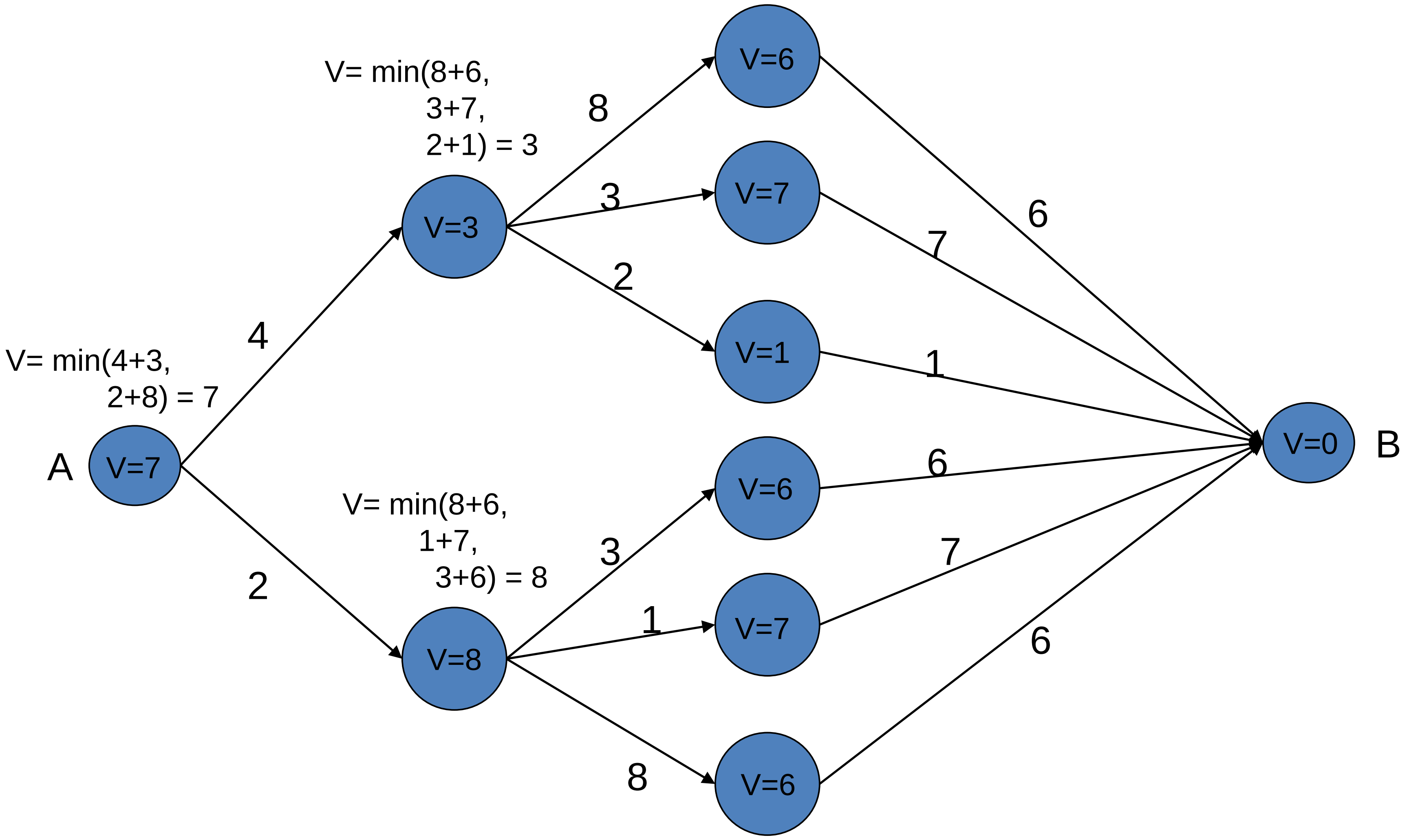
Find Shortest Path from A to B (using DP)



Find Shortest Path from A to B (using DP)

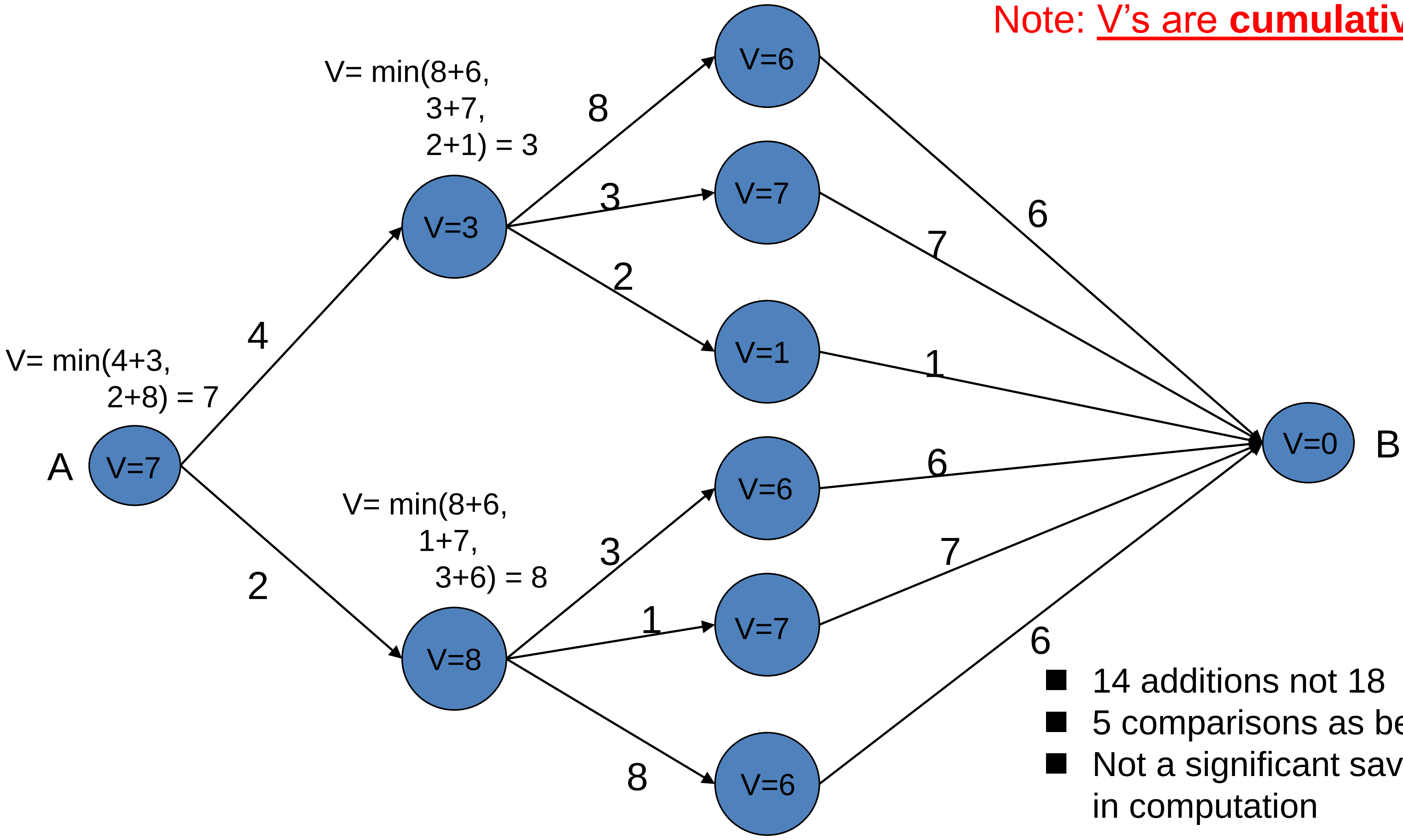


Find Shortest Path from A to B (using DP)



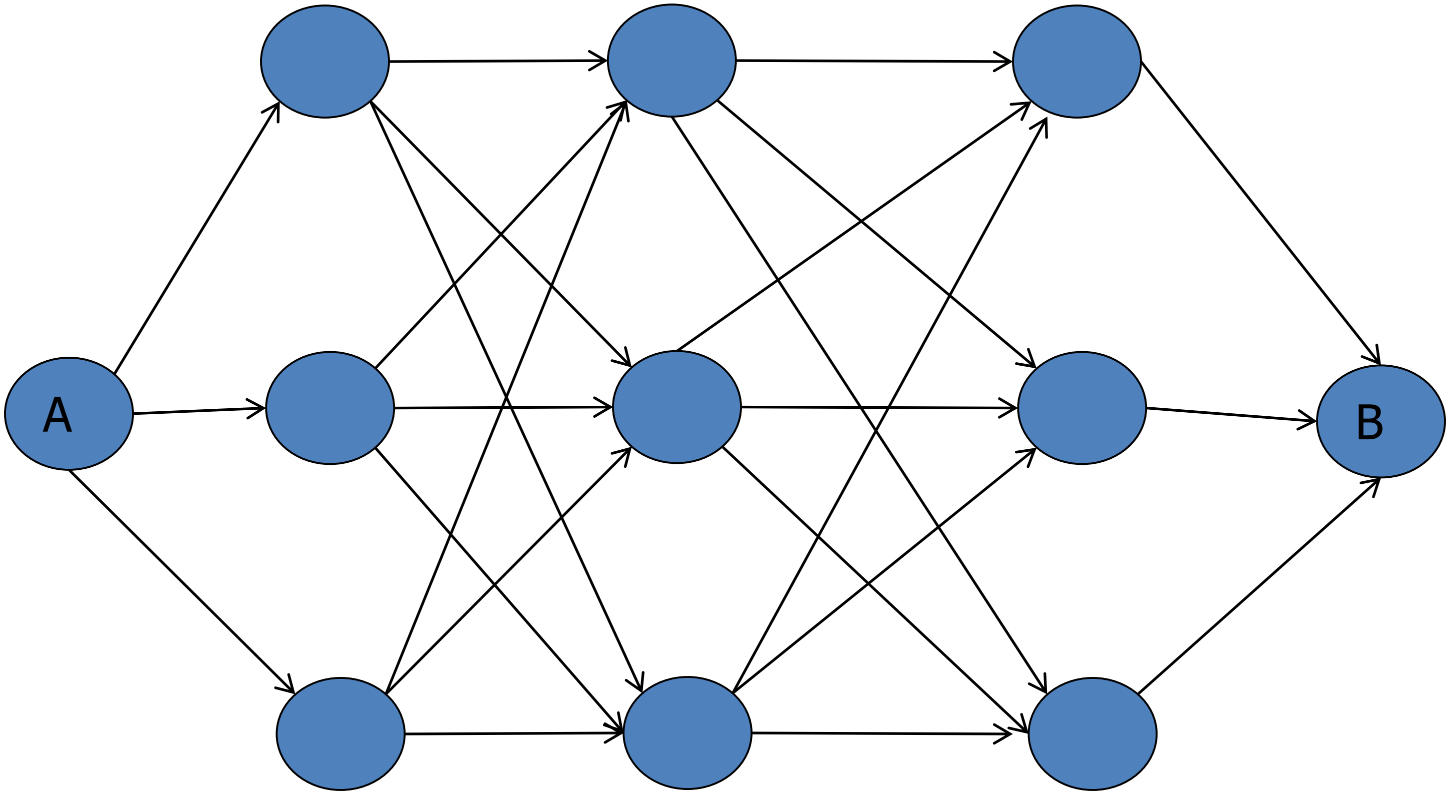
Find Shortest Path from A to B (using DP)

Note: V's are **cumulative**



- 14 additions not 18
- 5 comparisons as before
- Not a significant saving in computation

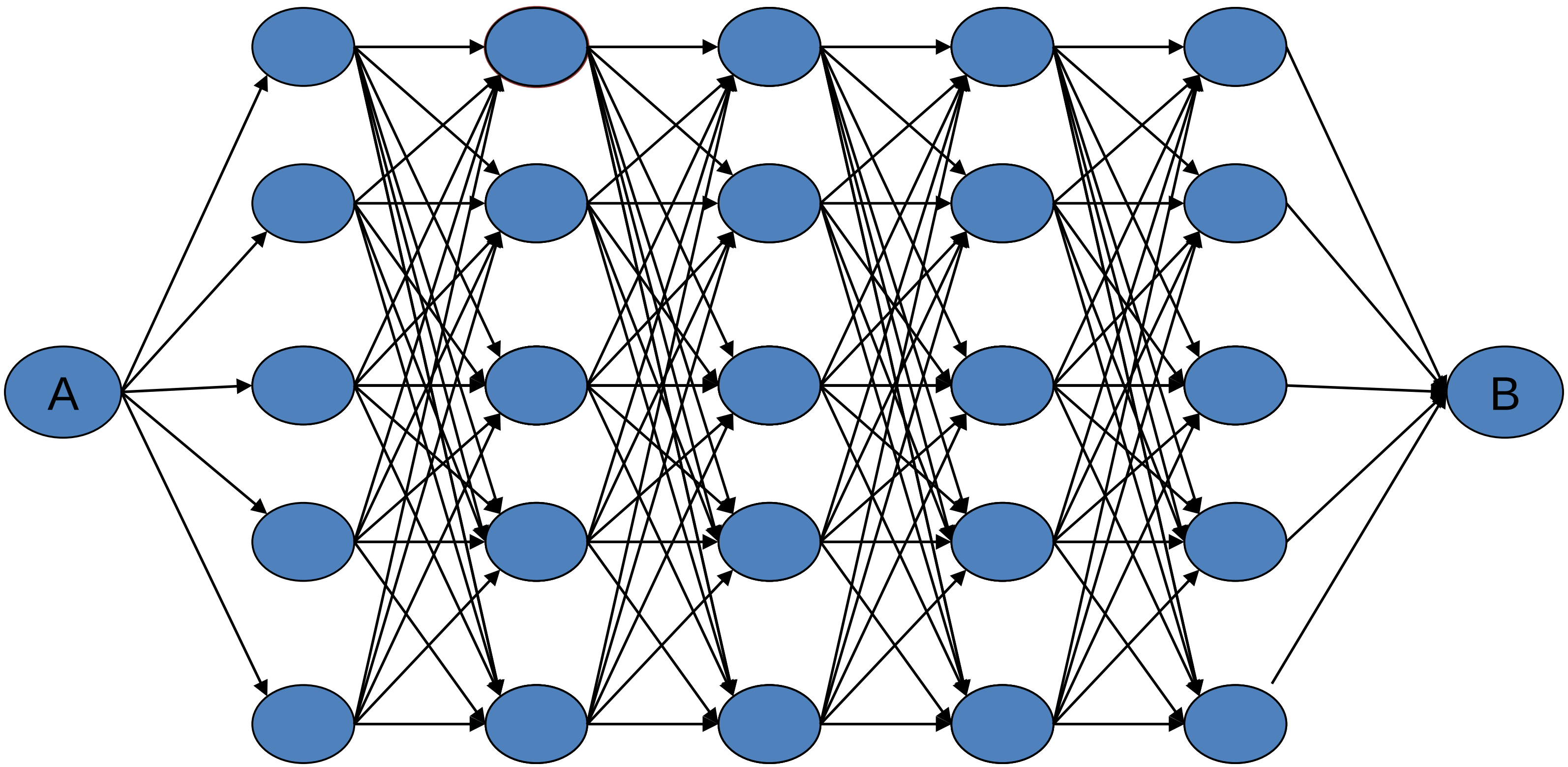
Another Example



Explicit enumeration
27*3= 81 additions
26 comparisons

Backward recursion
24 additions
13 comparisons

Another Example



Explicit enumeration
5^5 paths*5 additions per path=15625 additions
$5^5 - 1$ comparisons = 3124

Backward recursion
4*(25)+10=110 additions
20*4+1 comparisons = 81

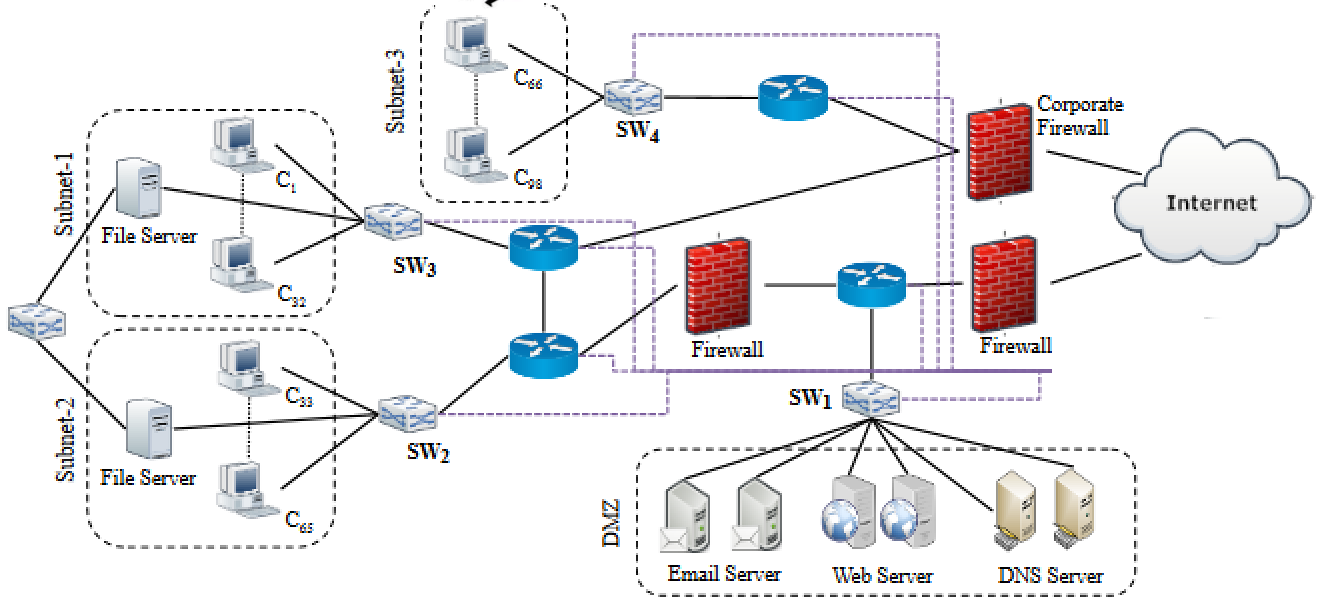
A significant saving in computation!!!

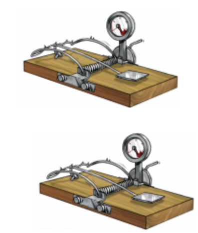
Backward Recursion

- Real world problems cannot be solved backwards because time flows forward
- So we need to estimate the value of the future states
- We estimate the value of the future states almost accurately by learning in a simulator which interacts with the environment
- We make random decisions initially and learn from those and then become greedy eventually by making only the best decisions

Monitoring an Organization’s Network

Malicious Activity

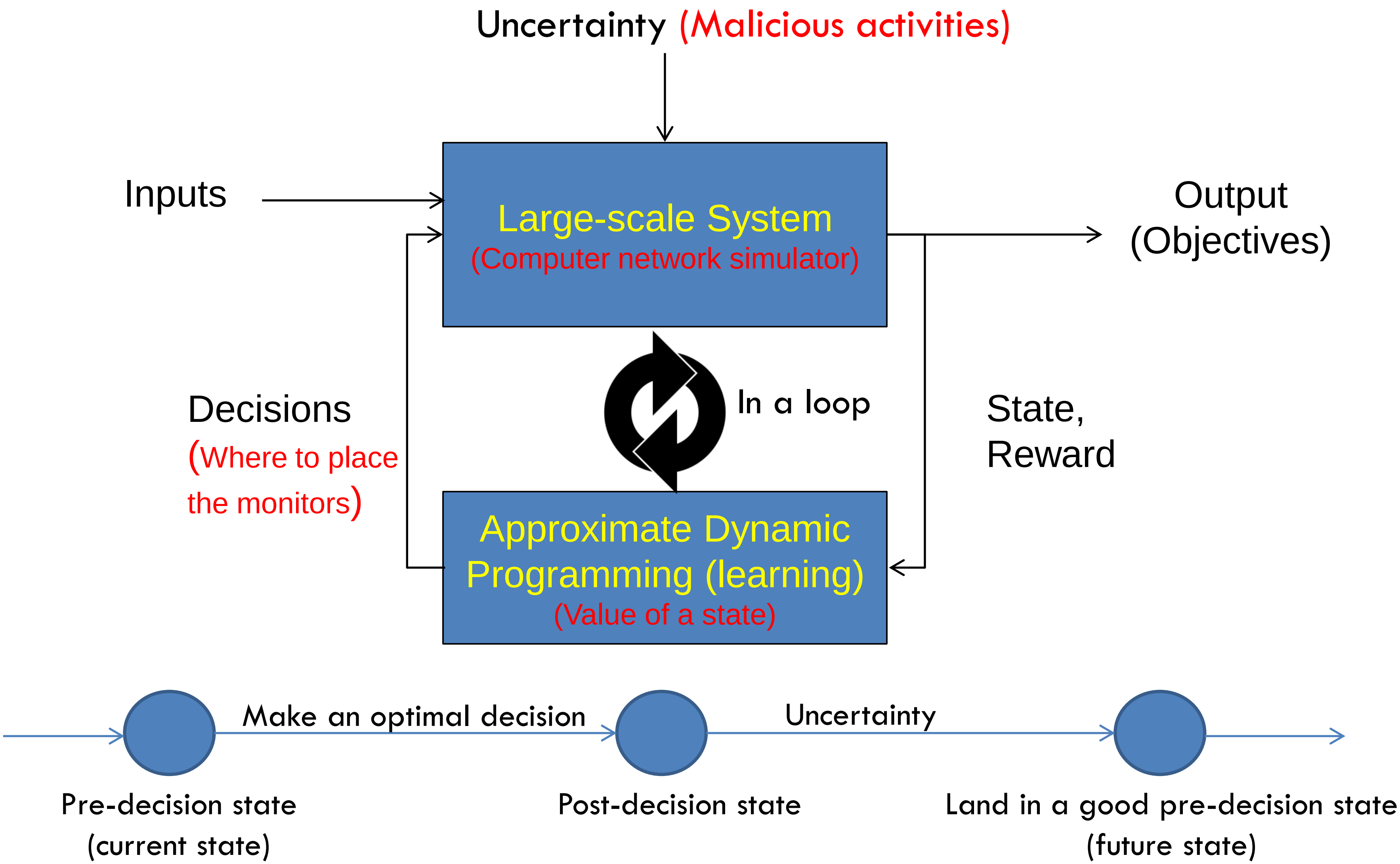




[Venkatesan et al. 2017]

Monitors/Honeypots

Simulation-based Optimization



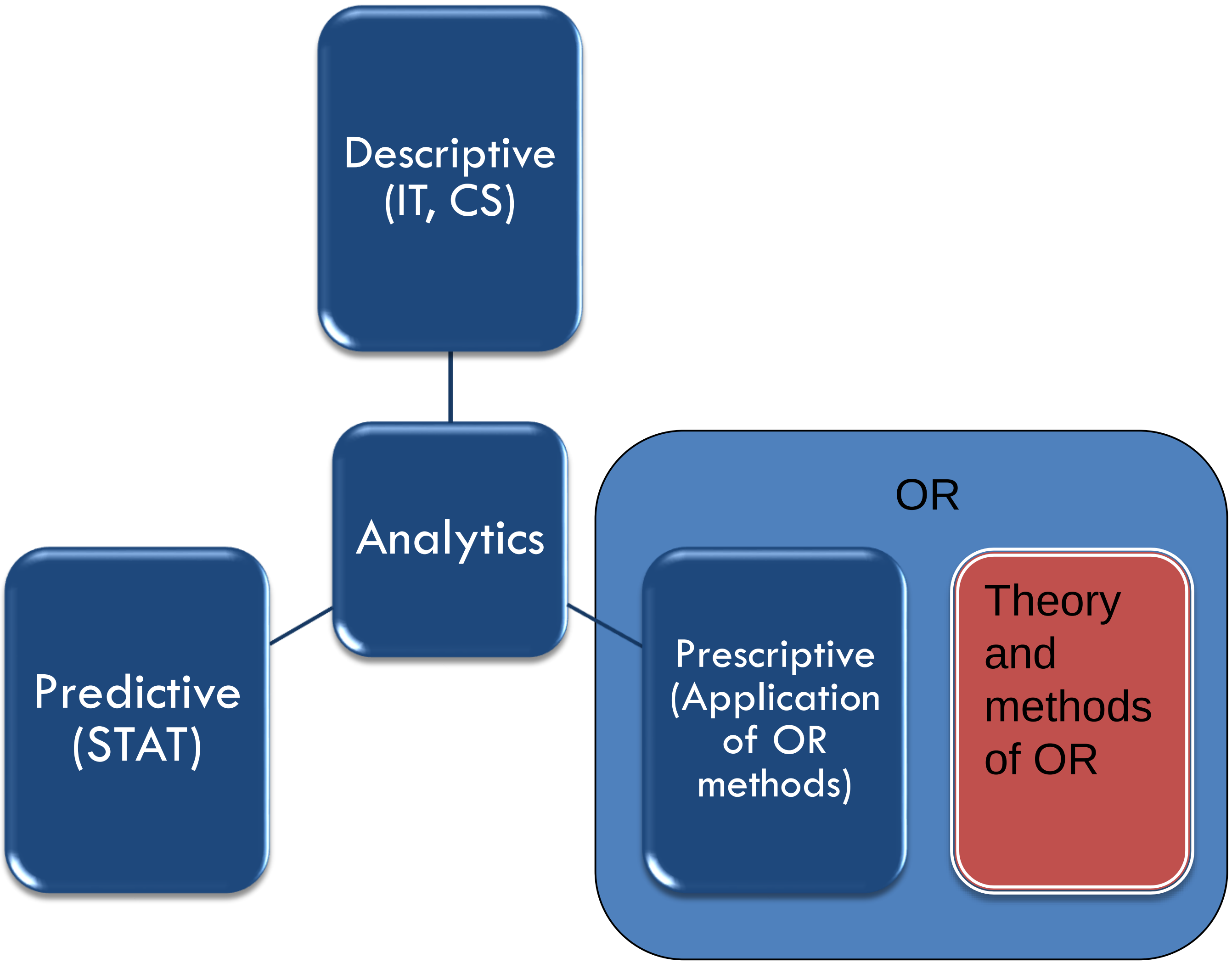
Dynamic Programming
for
Sequential Decision Making (over time)

is based on the idea that
we want to **move from one good state** of the system **to another**
by
making a near-optimal decision
in the presence of uncertainty

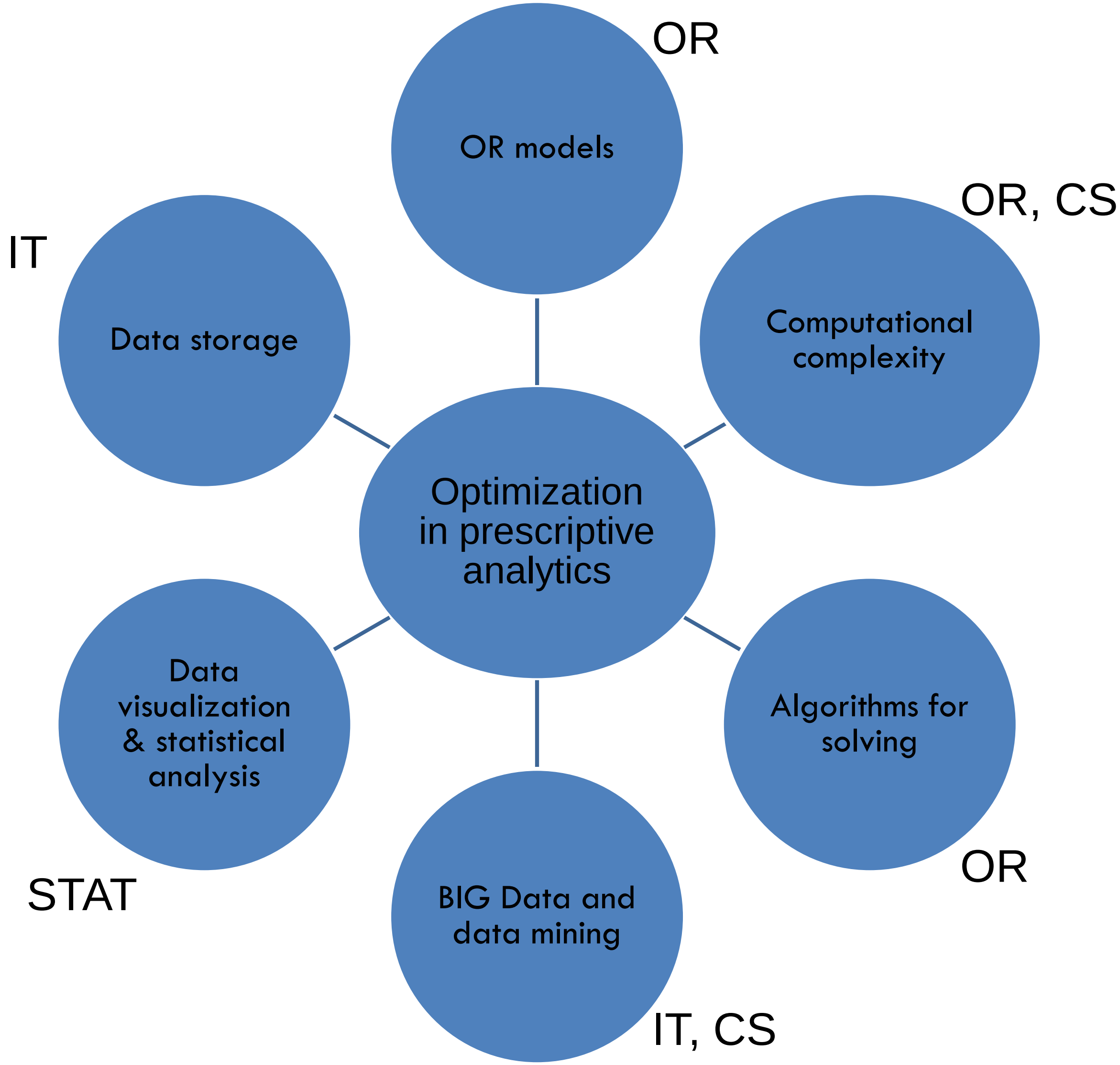
In large scale problems, the above is achieved via reinforcement learning (**approximate dynamic programming**) that entails only an interaction with the environment in a model-free setting

In Summary

Analytics and Operations Research (OR)



Optimization in Prescriptive Analytics



Big Data Decision Making Problems

- Understand characteristics of the data, linear/non-linear, deterministic/stochastic, static/dynamic (frequency of collection).
- Beware of:
  - Myopic policies
  - Exhaustive enumeration of all solution
  - Computational complexity

Computational Aspects

- LP - software has been developed. It has been widely researched.
- IP and NLP are more difficult to solve than LP (software exists). Well researched.
- Large-scale Stochastic DP (ADP in particular) is not well researched and is a newer field (no software, have to write the code).
  - Computationally far difficult than LP, IP, NLP but we are getting better with faster computers.
  - However, ADP is the only route for near-optimally solving some of the toughest DYNAMIC optimization problems in real-world.
    - Particularly for sequential decision making every few seconds in a fast changing and uncertain environment.
  - If you solve it, PATENT IT!!!

Main Take Away for Next Class

- Value function V is cumulative.
- When making a decision sequentially over time (dynamic programming):
  - Sum: **cost/reward of making the decision** with

    value of the estimated future state that the decision brings you to
- In this course, we solve optimally.
  - In OR 774 and in real-world problems, we strive of near-optimal (good enough) solutions.
- We will use Matlab and Excel to solve DP problems, however prior knowledge of Matlab or Excel use is not required.

Questions, Discussion, and Feedback

Thank you!

Contact Info:
Ankit Shah
ashah20@gmu.edu