IART - Artificial Intelligence

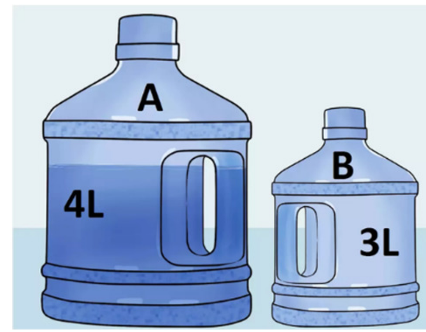
Exercise 1: Formulation of Search Problems

Luís Paulo Reis

LIACC – Artificial Intelligence and Computer Science Lab.
DEI/FEUP – Informatics Engineering Department, Faculty of Engineering of the University of Porto, Portugal
APPIA – Portuguese Association for Artificial Intelligence

Exercise: Bucket Filling Problem

Two buckets, with capacities c1 (ex: 4 liters) and c2 (ex: 3 liters), respectively, are initially empty. The buckets have no intermediate markings.



The only operations you can perform are:

- empty a bucket
- fill (completely) a bucket
- pour one bucket into the other until the second is full
- pour one bucket into the other until the first is empty

The objective is to determine which operations to carry out so that the first bucket contains n liters (example: 2 liters)?

a) Formulate the Problem as a search problem

b) Solve the problem through a tree search

Search Problem Formulation

- **State Representation**
  - Typically a combination of some variables, arrays and matrixes
  - In this case obviously only two variables needed
- **Initial (Current) State**
- **Objective Test (defines the desired states)**
  - Typically define the function

    bool objective_test(State)
  - In this case we have a clear objective state

Search Problem Formulation

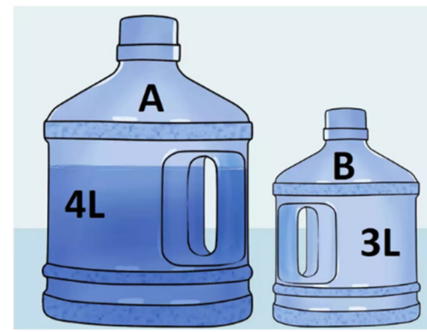
- **Operators (Name, Preconditions, Effects, cost)**
  - For each operator define a name for it and define the functions:

    bool preconditions(State, Operator)

    and
    State effects(State, Operator)
    and the cost of each operator
- **Solution Cost**
  - Typically the sum of the cost of the operators used to get from the initial state to an objective state

Exercise: Bucket Filling Problem

Two buckets, with capacities c1 (ex: 4 liters) and c2 (ex: 3 liters), respectively, are initially empty. The buckets have no intermediate markings.



The only operations you can perform are:

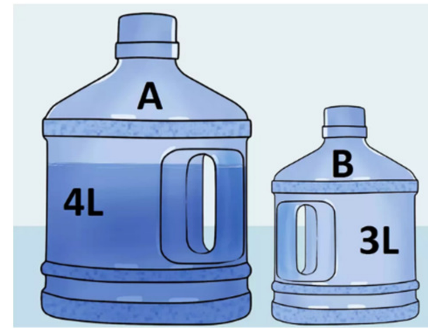
- empty a bucket
- fill (completely) a bucket
- pour one bucket into the other until the second is full
- pour one bucket into the other until the first is empty

The objective is to determine which operations to carry out so that the first bucket contains n liters (example: 2 liters)?

a) Formulate the Problem as a search problem

b) Solve the problem through a tree search

Bucket Filling Problem Formulation



- **State Representation:**

  ???

- **Initial State:**

  ???

- **Objective State:**

  ???

- **Operators:**

  ???

Bucket Filling Problem Formulation

- **State Representation:**

  [W1/W2]   W1:0..4; W2:0..3

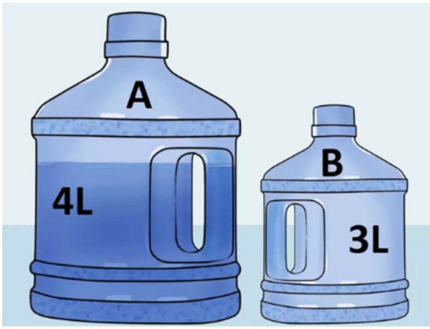
- **Initial State:**

  [0/0]

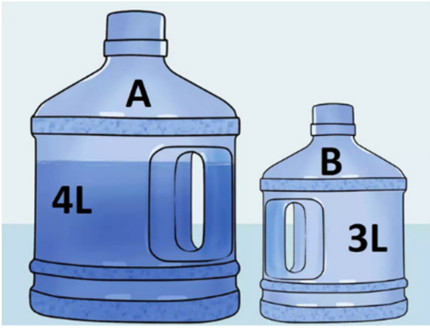
- **Objective State:**

  [2/_] or in general case [NL/_]

- **Operators:**



Bucket Filling Problem Formulation



- **State Representation:**

  [W1/W2]   W1:0..4; W2:0..3

- **Initial State:**

  [0/0]

- **Objective State:**

  [2/_] or in general case [NL/_]

- **Operators (3 possibilities):**

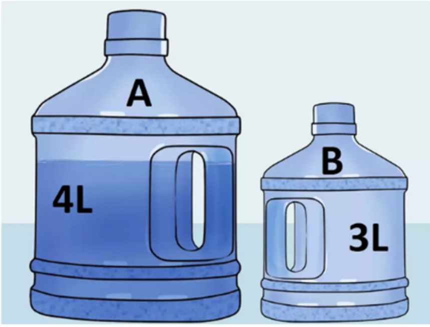
  Emp1, Emp2, Fill1, Fill2, Pour12a, Pour 12b, Pour21a, Pour21b

  Emp1, Emp2, Fill1, Fill2, Pour12, Pour21

  Emp(x), Fill(x), Pour(x,y) - could be good for multiple buckets

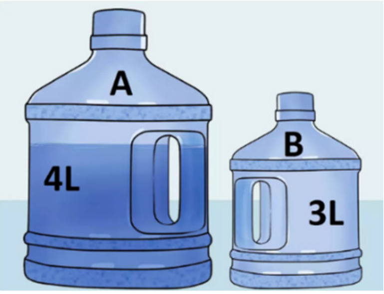
Bucket Filling Problem Formulation

- **Operators (1st possibility):**
  - Emp1 – empty bucket 1
  - Emp2 – empty bucket 2
  - Fill1 – fill bucket 1
  - Fill2 – fill bucket 2
  - Pour12a – pour bucket 1 to 2 until 2 is full
  - Pour12b – pour bucket 1 to 2 until 1 is empty
  - Pour21a – pour bucket 2 to 1 until 1 is full
  - Pour21b – pour bucket 2 to 1 until 2 is empty



Typically there are lots of modelling possibilities for the state and operators

Bucket Filling Problem Formulation

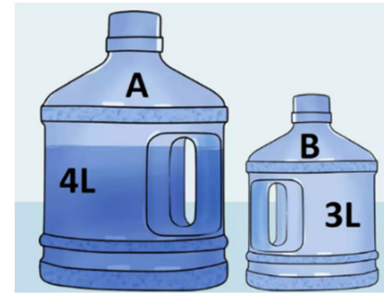


Operators (using possibility 1):

Emp1, Emp2, Fill1, Fill2, Pour12a, Pour12b, Pour21a, Pour21b

Name PreCond Effects Cost

Bucket Filling Problem Formulation

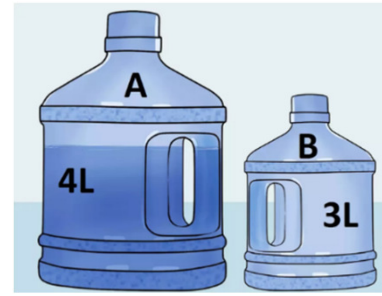


Operators (using possibility 1):

Emp1, Emp2, Fill1, Fill2, Pour12a, Pour12b, Pour21a, Pour21b

Name	PreCond	Effects	Cost
Emp1	W1>0	W1=0	1
Fill1	…		

Bucket Filling Problem Formulation

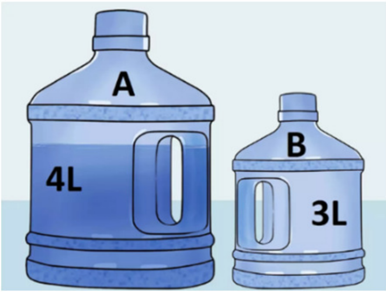


Operators (using possibility 1):

Emp1, Emp2, Fill1, Fill2, Pour12a, Pour12b, Pour21a, Pour21b

Name	PreCond	Effects	Cost
Emp1	W1>0	W1=0	1
Emp2	W2>0	W2=0	1
Fill1	W1<4	W1=4	1
Fill2	W2<3	W2=3	1
Po12a	…		

Bucket Filling Problem Formulation

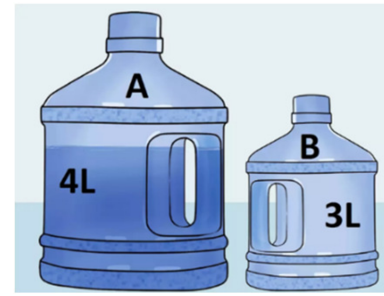


Operators (using possibility 1):

Emp1, Emp2, Fill1, Fill2, Pour12a, Pour12b, Pour21a, Pour21b

Name	PreCond	Effects	Cost
Emp1	W1>0	W1=0	1
Emp2	W2>0	W2=0	1
Fill1	W1<4	W1=4	1
Fill2	W2<3	W2=3	1
Po12a	W1>0/\W2<3/\W1>=3-W2	W1=W1-(3-W2);W2=3	1
Po12b	…		
Po21a	…		
Po21b	…		

Bucket Filling Problem Formulation

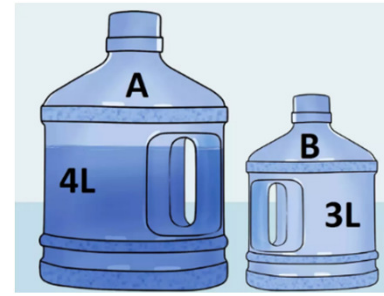


Operators (using moddeling 1):

Emp1, Emp2, Fill1, Fill2, Pour12a, Pour12b, Pour21a, Pour21b

Name	PreCond	Effects	Cost
Emp1	W1>0	W1=0	1
Emp2	W2>0	W2=0	1
Fill1	W1<4	W1=4	1
Fill2	W2<3	W2=3	1
Po12a	W1>0/\W2<3/\W1>=3-W2	W1=W1-(3-W2);W2=3	1
Po12b	W1>0/\W2<3/\W1<3-W2	W2=W1+W2;W1=0	1
Po21a	W2>0/\W1<4/\W2>=4-W1	W2=W2-(4-W1);W1=4	1
Po21b	W2>0/\W1<4/\W2<4-W1	W1=W1+W2;W2=0	1

Bucket Filling Problem Formulation



General Case (capacities C1, C2)

Operators (using moddeling 1):

Emp1, Emp2, Fill1, Fill2, Pour12a, Pour12b, Pour21a, Pour21b

Name	PreCond	Effects	Cost
Emp1	W1>0	W1=0	1
Emp2	W2>0	W2=0	1
Fill1	W1<C1	W1=C1	1
Fill2	W2<C2	W2=C2	1
Po12a	W1>0/\W2<C2/\W1>=C2-W2	W1=W1-(C2-W2);W2=C2	1
Po12b	W1>0/\W2<C2/\W1<C2-W2	W2=W1+W2;W1=0	1
Po21a	W2>0/\W1<C1/\W2>=C1-W1	W2=W2-(C1-W1);W1=C1	1
Po21b	W2>0/\W1<C1/\W2<C1-W1	W1=W1+W2;W2=0	1

Bucket Filling Problem Formulation

Other possible Costs: C2 -Water Poured from tap ; C3 - Water wasted; C4 – Weight carried

Operators (using possibility 1):

Emp1, Emp2, Fill1, Fill2, Pour12a, Pour12b, Pour21a, Pour21b

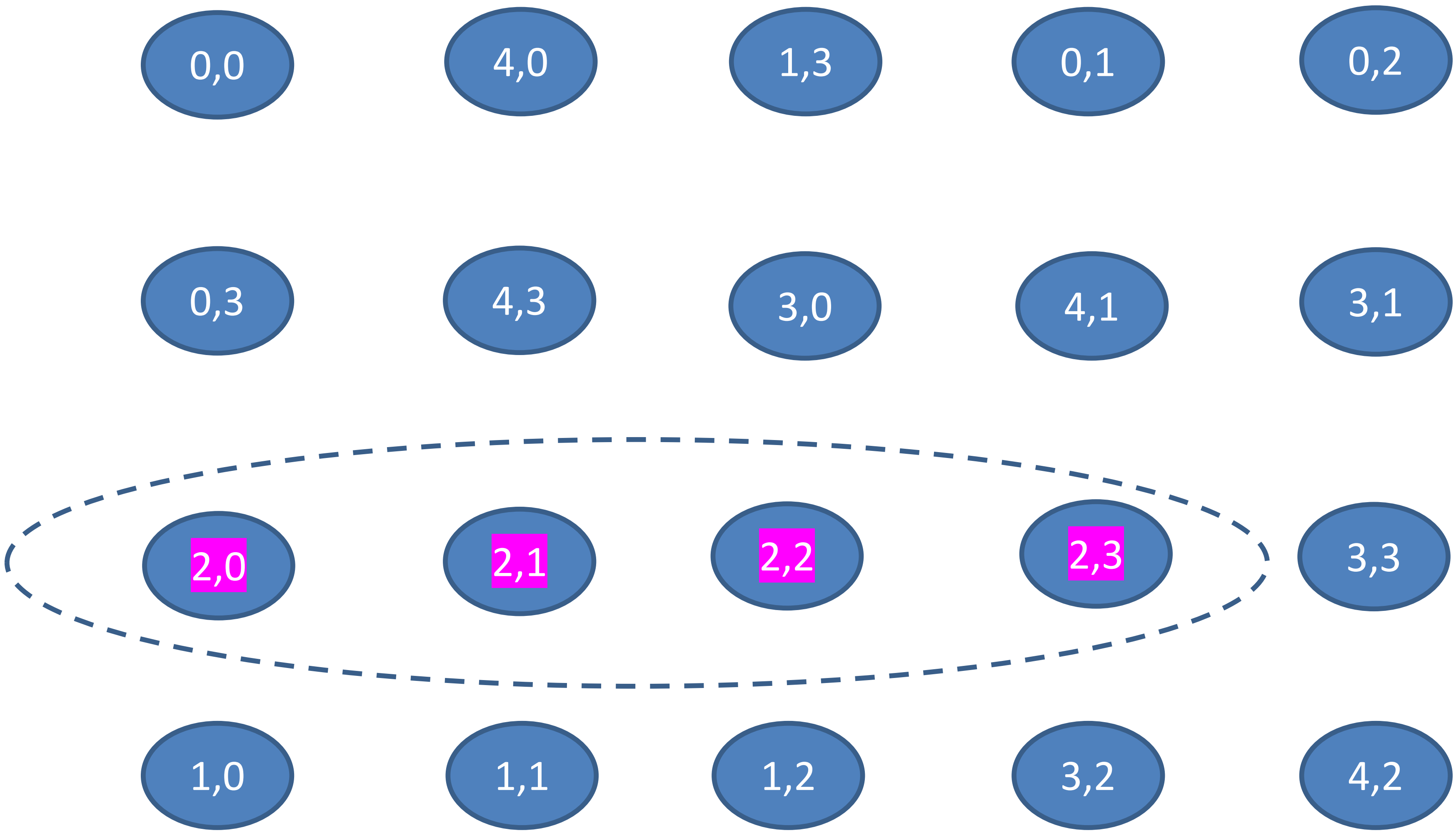
Name	PreCond	Effects	**C2**	**C3**
Emp1	W1>0	W1=0	**0**	**W1**
Emp2	W2>0	W2=0	**0**	**W2**
Fill1	W1<C1	W1=C1	**C1-W1**	**0**
Fill2	W2<C2	W2=C2	**C2-W2**	**0**
Po12a	W1>0/\W2<C2/\W1>=C2-W2	W1=W1-(C2-W2);W2=C2	**0**	**0**
Po12b	W1>0/\W2<C2/\W1<C2-W2	W2=W1+W2;W1=0	**0**	**0**
Po21a	W2>0/\W1<C1/\W2>=C1-W1	W2=W2-(C1-W1);W1=C1	**0**	**0**
Po21b	W2>0/\W1<C1/\W2<C1-W1	W1=W1+W2;W2=0	**0**	**0**

State Space

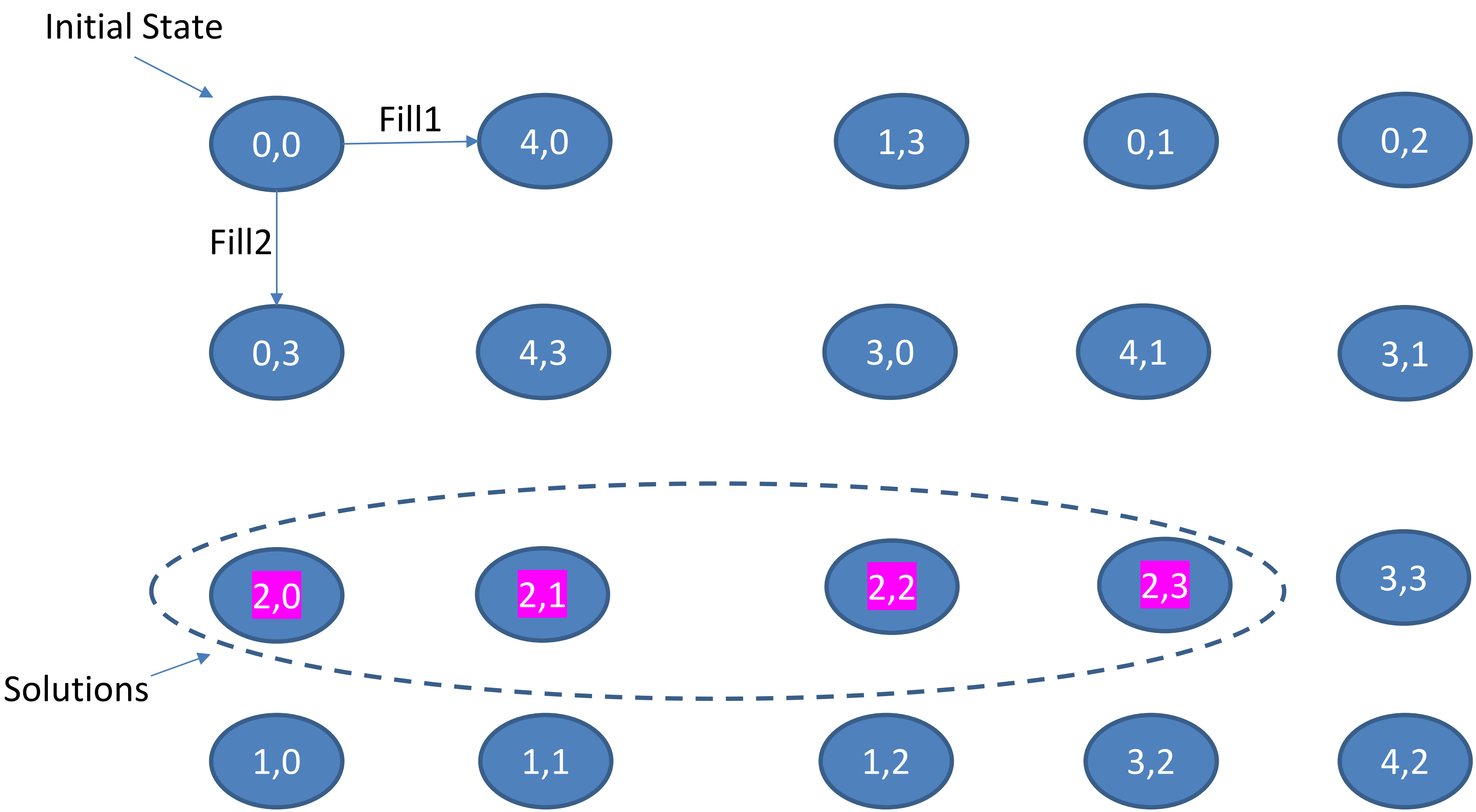
- What is the State Space Size for this problem with two buckets of capacity 4, 3?

State Space

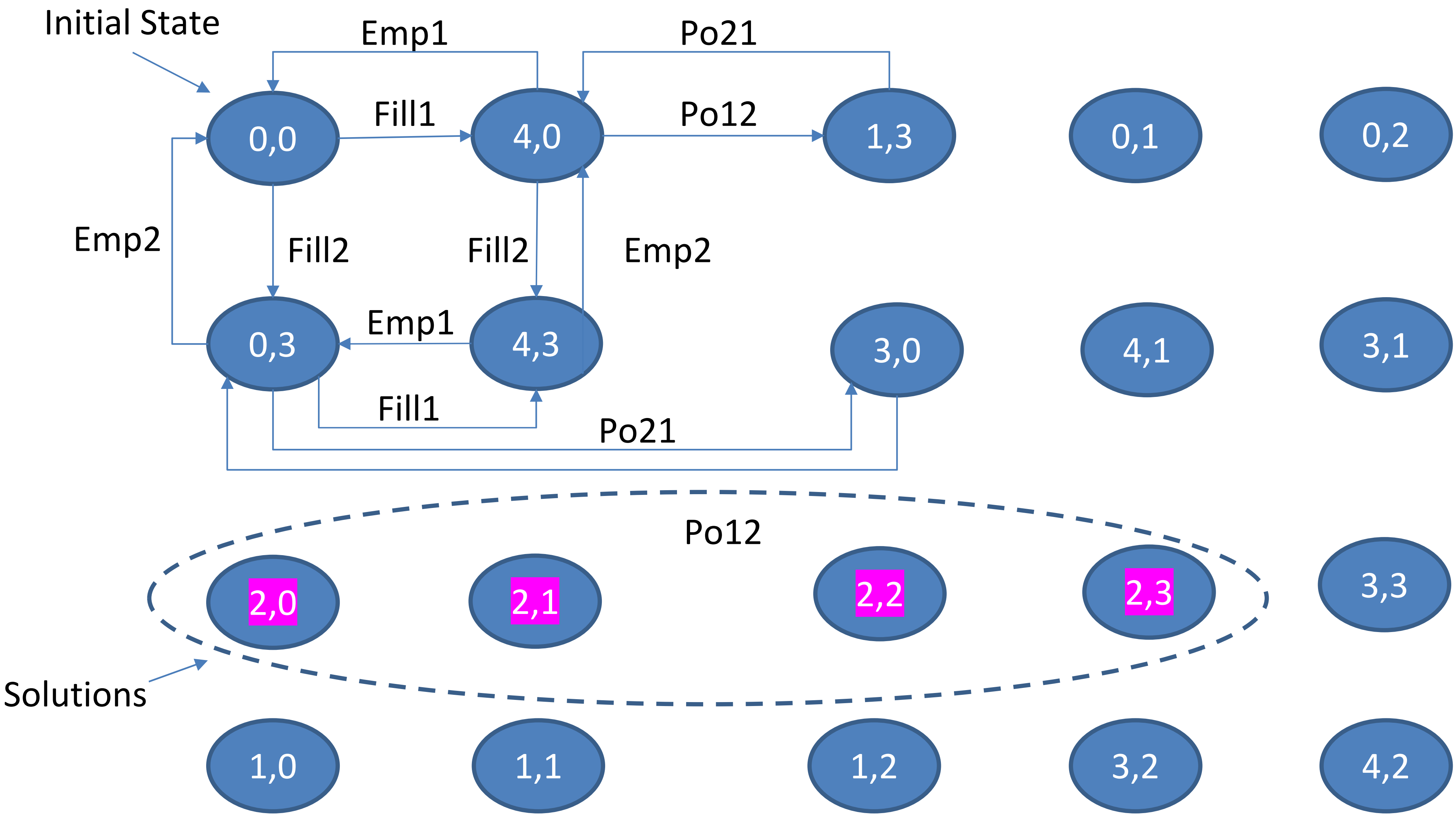
- What is the State Space Size for this problem with two buckets of capacity 4, 3? Size = 5*4 = 20 states



State Space


Initial State
0,0
Fill1
4,0
1,3
0,1
0,2
Fill2
0,3
4,3
3,0
4,1
3,1
2,0
2,1
2,2
2,3
3,3
Solutions
1,0
1,1
1,2
3,2
4,2

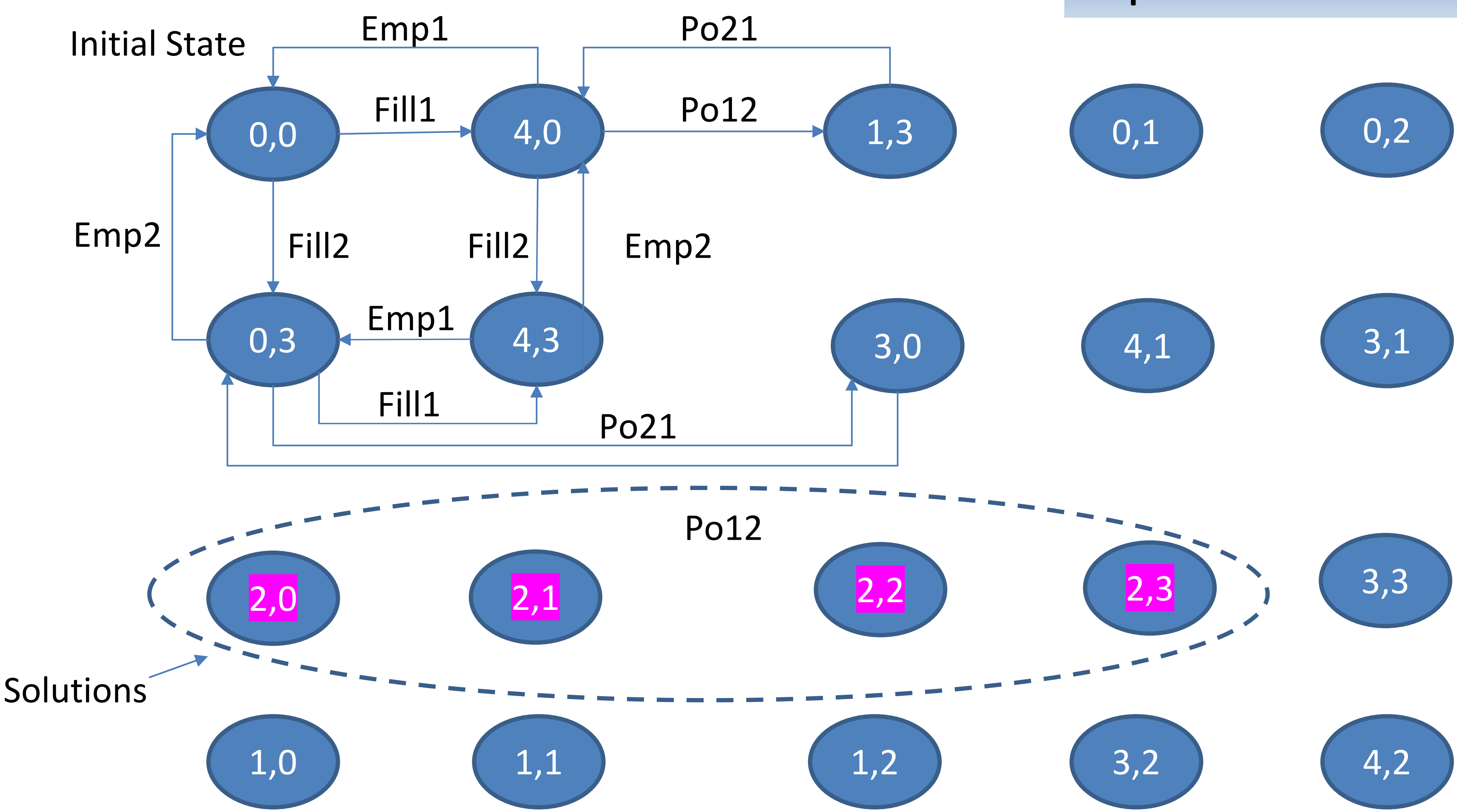
State Space



Initial State
Emp1
Po21
0,0
Fill1
4,0
Po12
1,3
0,1
0,2
Emp2
Fill2
Fill2
Emp2
0,3
Emp1
4,3
3,0
4,1
3,1
Fill1
Po21
Po12
2,0
2,1
2,2
2,3
3,3
Solutions
1,0
1,1
1,2
3,2
4,2

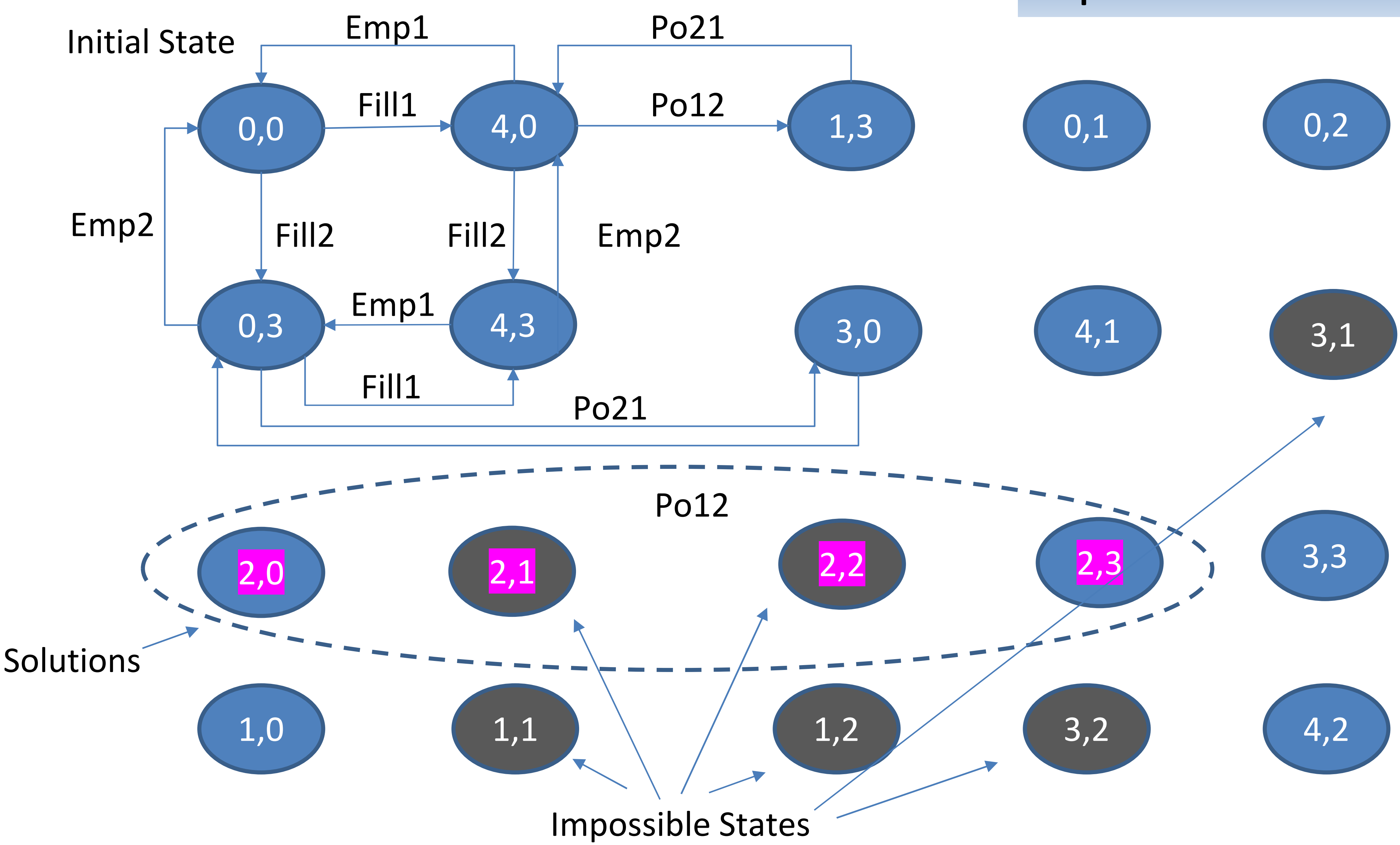
State Space

Exercise: Complete the connections to get the 2 optimal solutions

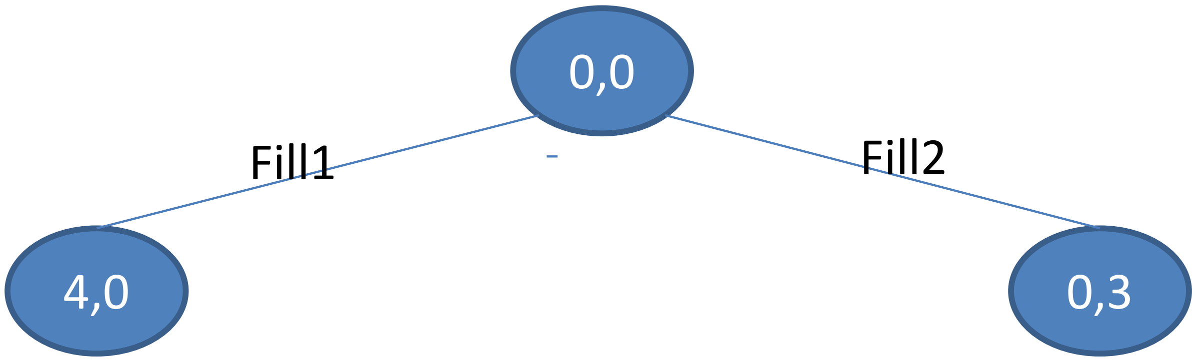


State Space

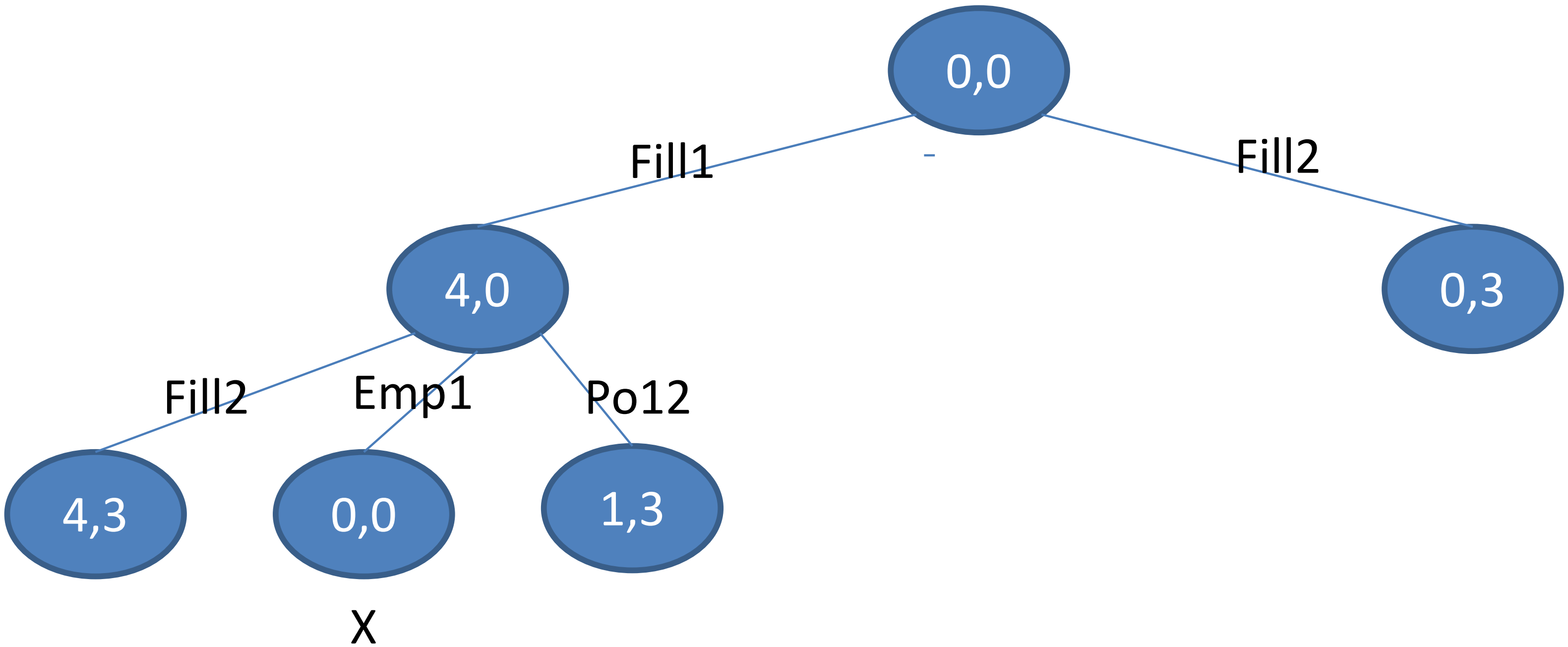
Exercise: Complete the connections to get the 2 optimal solutions



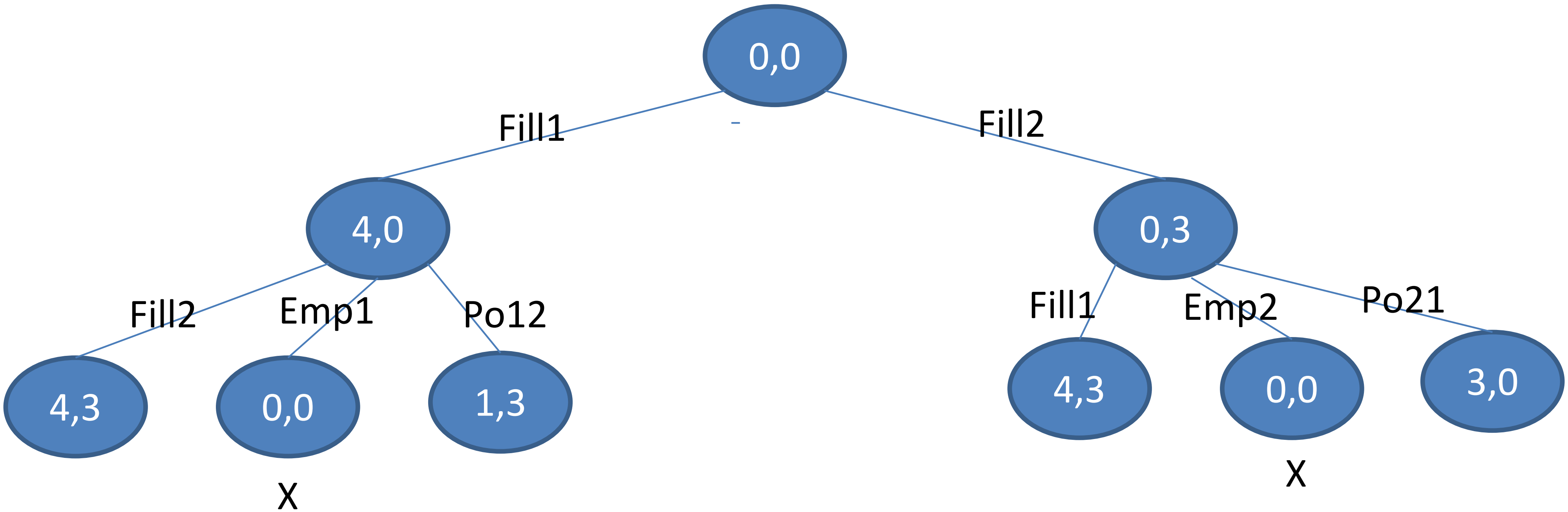
Breadth First Search



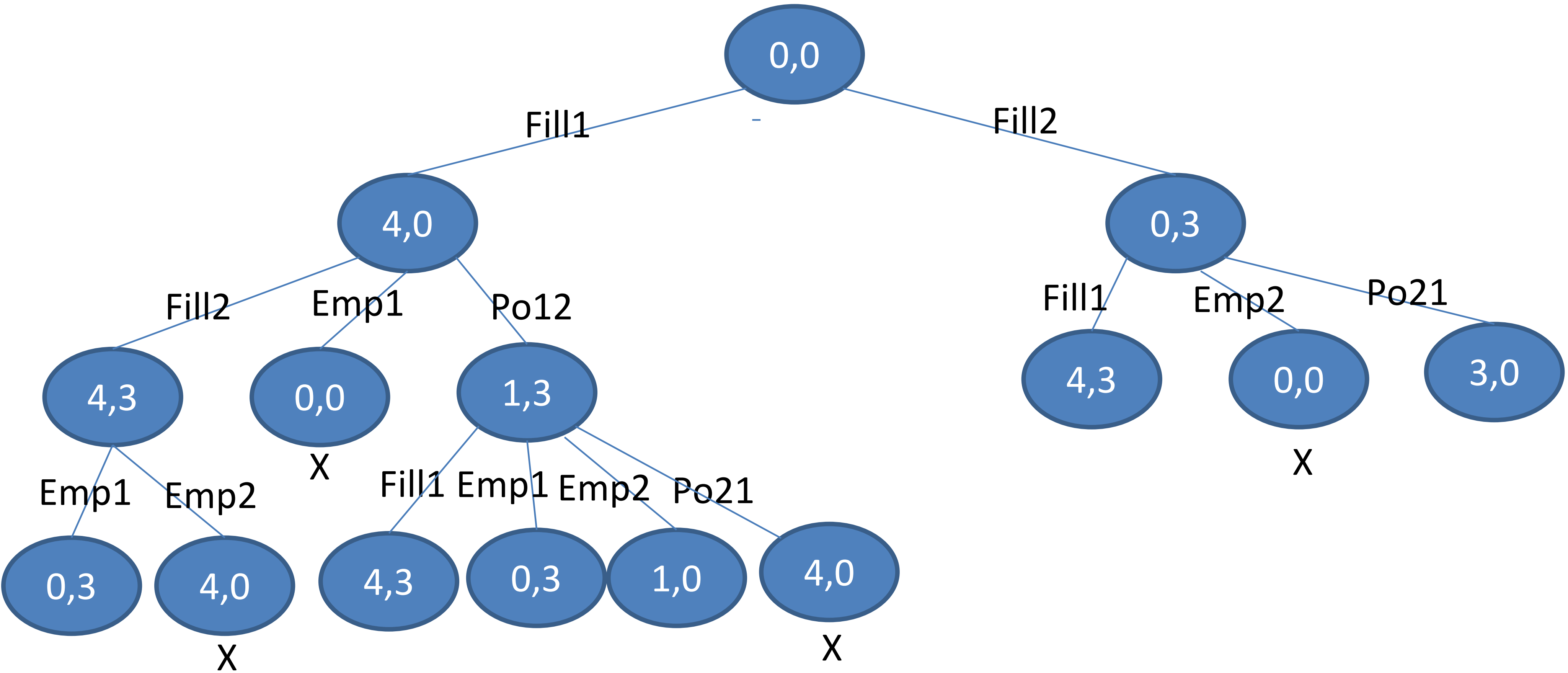
Breadth First Search



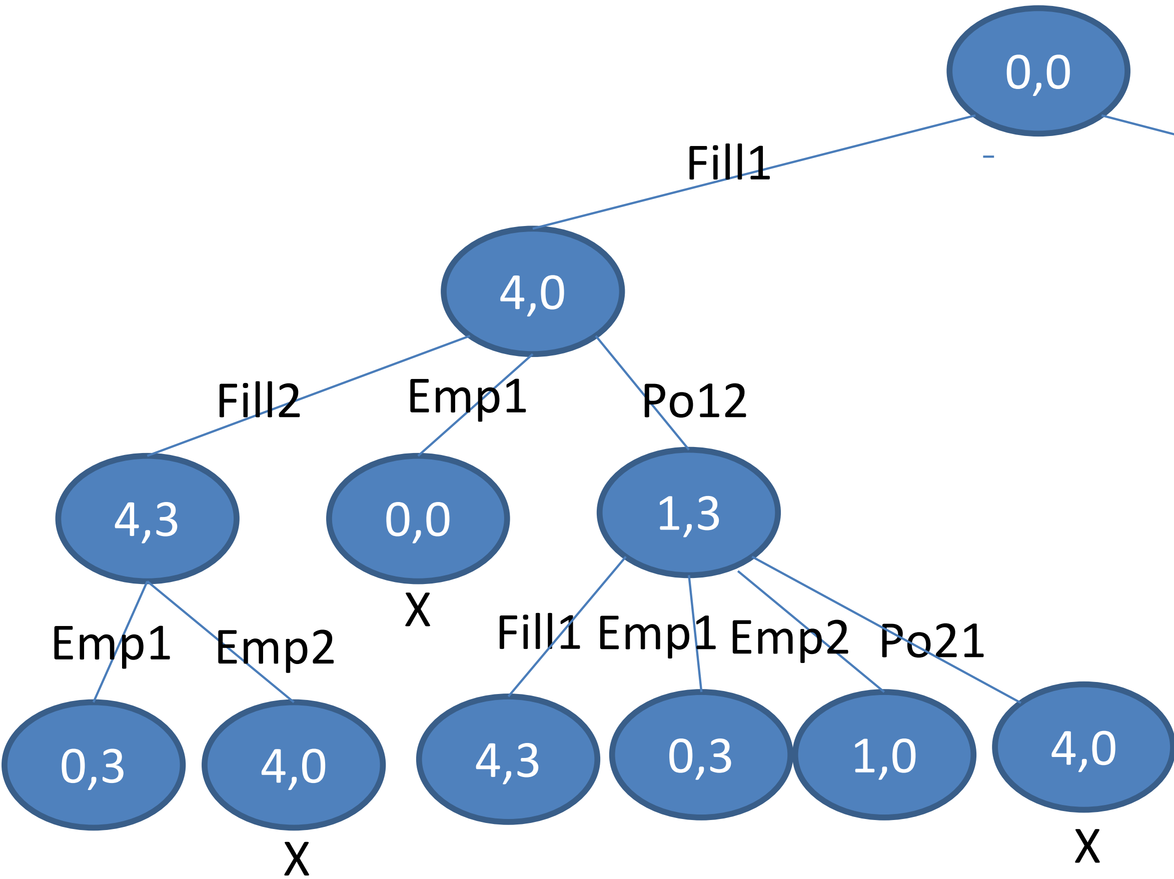
Breadth First Search



Breadth First Search



Breadth First Search



Exercise: Complete the tree to get the 2 optimal solutions

Exercise: Create a simple code to solve the problem with BFS

Breadth First Search

```
from collections import deque

def breadth_first_search(initial_state, goal_state_func, operators_func):
  root = TreeNode(initial_state)   # create the root node in the search tree
  queue = deque([root])   # initialize the queue to store the nodes
  #visited = set([initial_state])   # to avoid loops –only in snall problems
  while queue:
    node = queue.popleft()   # get first element in the queue
    if goal_state_func(node.state):   return node    # check goal state
    for state in operators_func(node.state):   # go through next states
        #if state not in visited:   # to avoid loops
        # create tree node with the new state
        child_node = TreeNode(state=state, parent=node)
        # link child node to its parent in the tree
        node.add_child(child_node)
        # enqueue the child node
        queue.append(child_node)
        #visited.add(state)   # to avoid loops
  return None
```

Exercise: Missionaries and Cannibals

3 missionaries and 3 cannibals are on one side of the river with a boat that only takes 2 people.

The objective is to find a way to take the 6 to the other side of the river without ever leaving more cannibals than missionaries on one of the banks during the process!

a) Formulate this problem as a search problem, defining the representation of the state, initial state, the operators (and respective preconditions and effects), the objective test and the cost of the solution.

b) Solve the problem through a tree search

Missionaries/Cannibals Prob. Formulation

- **State Representation:**

  ???

- **Initial State:**

  ???

- **Objective State:**

  ???

- **Operators:**

  ???

Missionaries/Cannibals Prob. Formulation

- **State Representation:**

  [NM/NC/NB]   NM:0..3; NC:0..3 ; NB:0..1

  // Number of missionaries, cannibals and boats on the initial margin. On the other margin: 3-X

- **Initial State:**

  ???

- **Objective State:**

  ???

- **Operators:**

  ???

Missionaries/Cannibals Prob. Formulation

- **State Representation:**

  [NM/NC/NB]   NM:0..3; NC:0..3 ; NB:0..1

  // Number of missionaries, cannibals and boats on the initial margin. On the other margin: 3-X

- **Initial State:**

  [3/3/1] (all missionaries, cannibals and boats at first margin)

- **Objective State:**

  [0/0/_] or obviously [0/0/0] (all on the second margin)

- **Operators (3 possibilities):**

  ???

Missionaries/Cannibals Prob. Formulation

- **State Representation:**

  [NM/NC/NB]   NM:0..3; NC:0..3 ; NB:0..1

  // Number of missionaries, cannibals and boats on the initial margin. On the other margin: 3-X

- **Initial State:**

  [3/3/1] (all missionaries, cannibals and boats at first margin)

- **Objective State:**

  [0/0/_] or obviously [0/0/0] (all on the second margin)

- **Operators (3 possibilities):**

  MM1, CC1, MC1, M1, C1, MM2, CC2, MC2, M2, C2

  MM(Dir), CC(Dir), MC(Dir), M(Dir), C(Dir)

  Trip(NM,NC,Dir)

Missionaries/Cannibals Prob. Formulation

- **Operators**

  MM1, CC1, MC1, M1, C1 (trips from 1st to 2nd margin)

  MM2, CC2, MC2, M2, C2

Name	**PreCond**	**Effects**	**Cost**
MM1	NM>=2 /\ NB=1	NM=NM-2; NB=0	1

Missionaries/Cannibals Prob. Formulation

- **Operators**

  MM1, CC1, MC1, M1, C1, MM2, CC2, MC2, M2, C2

Name	**PreCond**	**Effects**	**Cost**
MM	NM>=2 /\ NB=1	NM=NM-2; NB=0	1

improve preconditions to assure that at the end of the move no missionaries are eaten?

Missionaries/Cannibals Prob. Formulation

- **Operators**

  MM1, CC1, MC1, M1, C1, MM2, CC2, MC2, M2, C2

Name	**PreCond**	**Effects**	**Cost**
MM	NM>=2 /\ NB=1	NM=NM-2; NB=0	1

Preconditions to assure that at the end of the move no missionaries are eaten?

NC<=NM-2 \/ NM=2 (missionaries not eaten at 1st margin)

and

3-NC <= 3-NM+2 (missionaries not eaten at 2nd margin)

...

Missionaries/Cannibals Prob. Formulation

- **Operators**

MM1, CC1, MC1, M1, C1, MM2, CC2, MC2, M2, C2

Name	PreCond	Effects	Cost
MM	NM>=2 /\ NB=1	NM=NM-2; NB=0	1

Preconditions to assure that at the end of the move no missionaries are eaten?

NC<=NM-2 \/ NM=2 (missionaries not eaten at 1st margin)

and

3-NC <= 3-NM+2 (missionaries not eaten at 2nd margin)

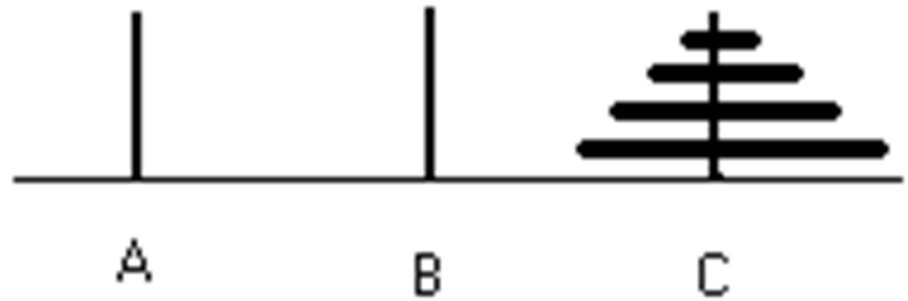
Exercise: Complete the problem formulation

Exercise: Solve manually with tree search the problem

Exercise: Create a simple code to solve the problem with BFS

Exercise: Hanoi Towers

a) Formulate the problem of the Towers of Hanoi as a search problem.



Notes:

- In this version of the problem, you have 3 towers (A, B and C) and 4 disks (D1 to D4).
- Initially the disks are in tower C and the objective is to transfer them to tower A.
- In each move, the player can move a disk from one tower to another tower, if he does not place that disk on a smaller disk.

b) Suppose that the number of disks and the number of towers can be different (n disks and m towers) and formulate this generic version of the problem as a search problem.

c) Solve the problem through a tree search

IART - Artificial Intelligence

Exercise 1: Search Problems

Luís Paulo Reis

LIACC – Artificial Intelligence and Computer Science Lab.
DEI/FEUP – Informatics Engineering Department, Faculty of Engineering of the University of Porto, Portugal
APPIA – Portuguese Association for Artificial Intelligence