

Artificial Intelligence

Note Title

7/7/2005

8-puzzle:

Given a sliding-block puzzle with 8 tiles, find a sequence of moves that will produce a goal configuration:

2		6
3	1	4
7	5	8

1	2	3
8		4
7	6	5

Search

: find a sequence of actions that can produce a desired outcome

Single-player: non-adversarial

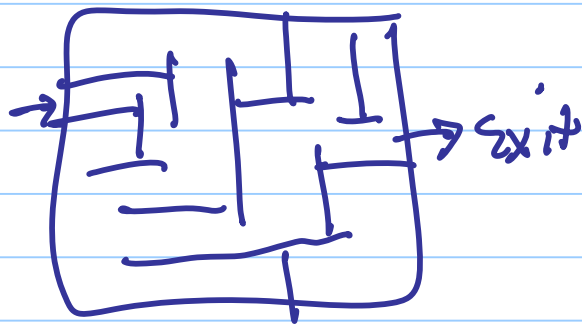
multiple players: adversarial

[goal: WIN
or make opponent
LOSE
or prevent own LOSS

probabilistic 1 + 2-person games : adversarial

↓
because player does not have total control

Maze Traversal :

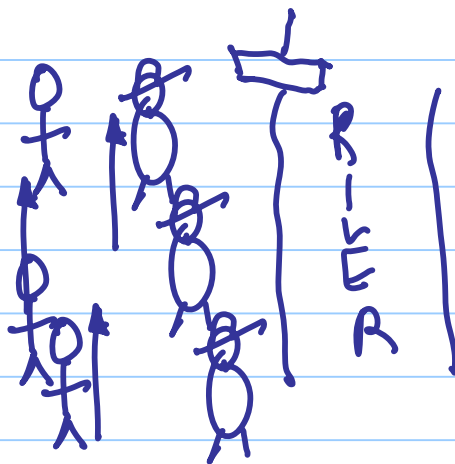


Initial Situation

Possible moves (actions)

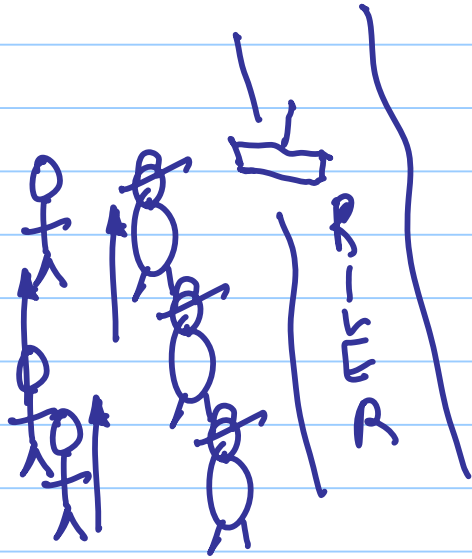
Goal Condition

Missionaries + Cannibals



- 3 M's + 3 C's on one bank of river
- 2 person boat
- if cannibals outnumber missionaries on one bank, they will kill them

Missionaries + Cannibals

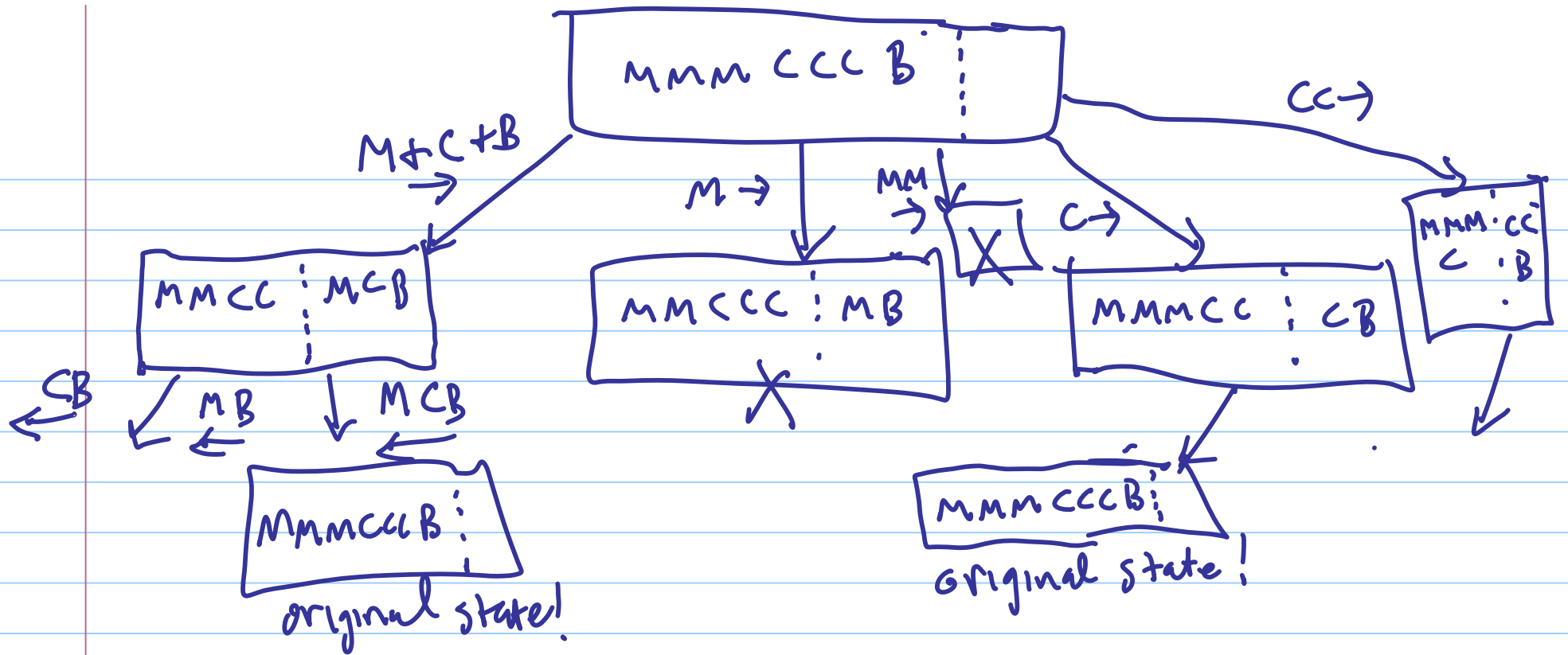


- 3 M's + 3 C's on one bank of river
- 2 person boat
- if cannibals outnumber missionaries on one bank, they will kill them

Goal - safely transport all 6 people across river

Initial State : (MMM CCC B)()

Goal : ()(MMM CCC B)



```
Backtrack( StateList ) // returns path to goal, or "Failure"
```

```
State = first( StateList )
```

```
if goal(State), return StateList
```

```
if deadEnd(State), return "Failure"
```

```
if State is a member of rest( StateList ), return "Failure"
```

```
Moves = all possible moves from State
```

```
for each m in Moves:
```

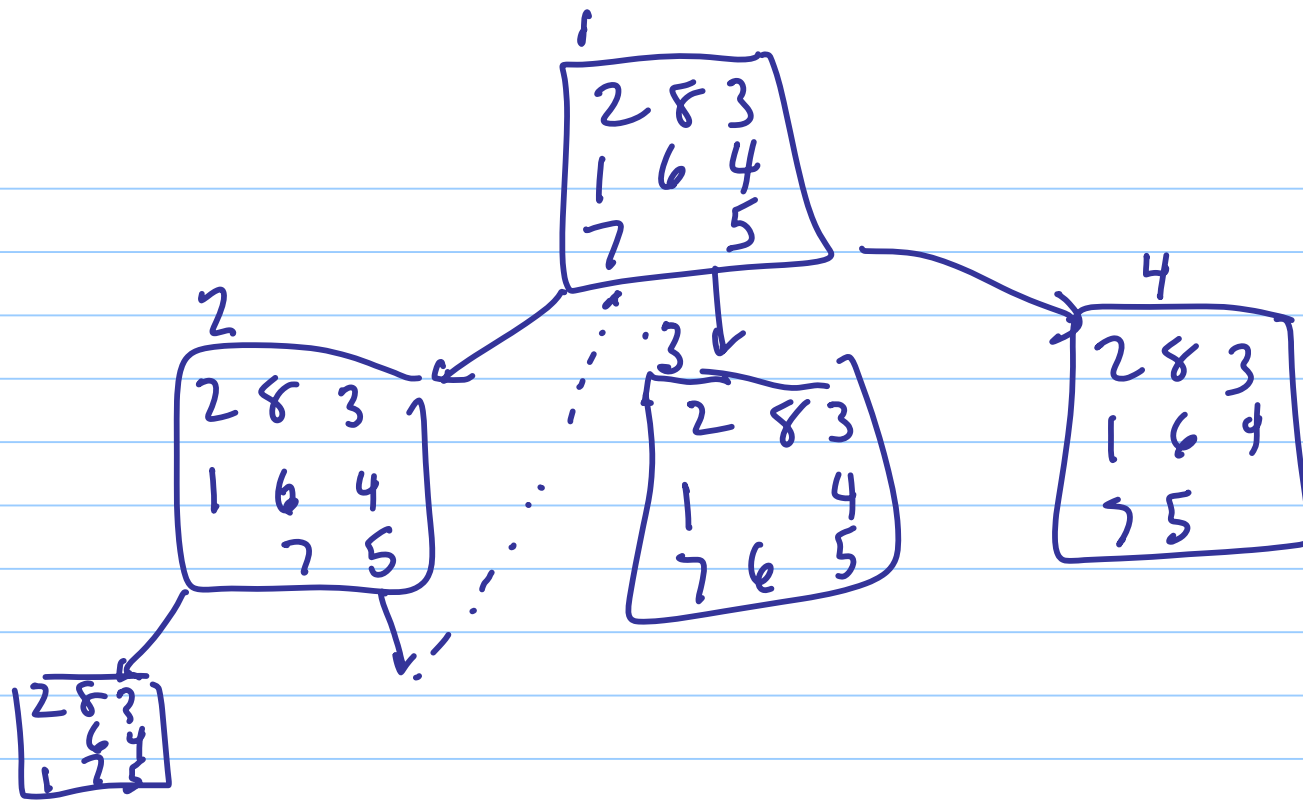
```
    nextState = applyMove(m,State)
```

```
    newStateList = nextState + StateList // add nextState to front of list
```

```
    path = Backtrack( newStateList )
```

```
    if path != "Failure", return path
```

```
return "Failure" // you only get here if all moves resulted in failure
```



Depth - first : always choose the most recent new node
+ expand it

Breadth - first : always choose nodes in "first come, first served" order

Best-first search:

Use a "ranking" to decide which node to expand next ("heuristic")

For example — let $f(\text{Board}) = \# \text{ tiles out of place}$
w.r.t. goal

state

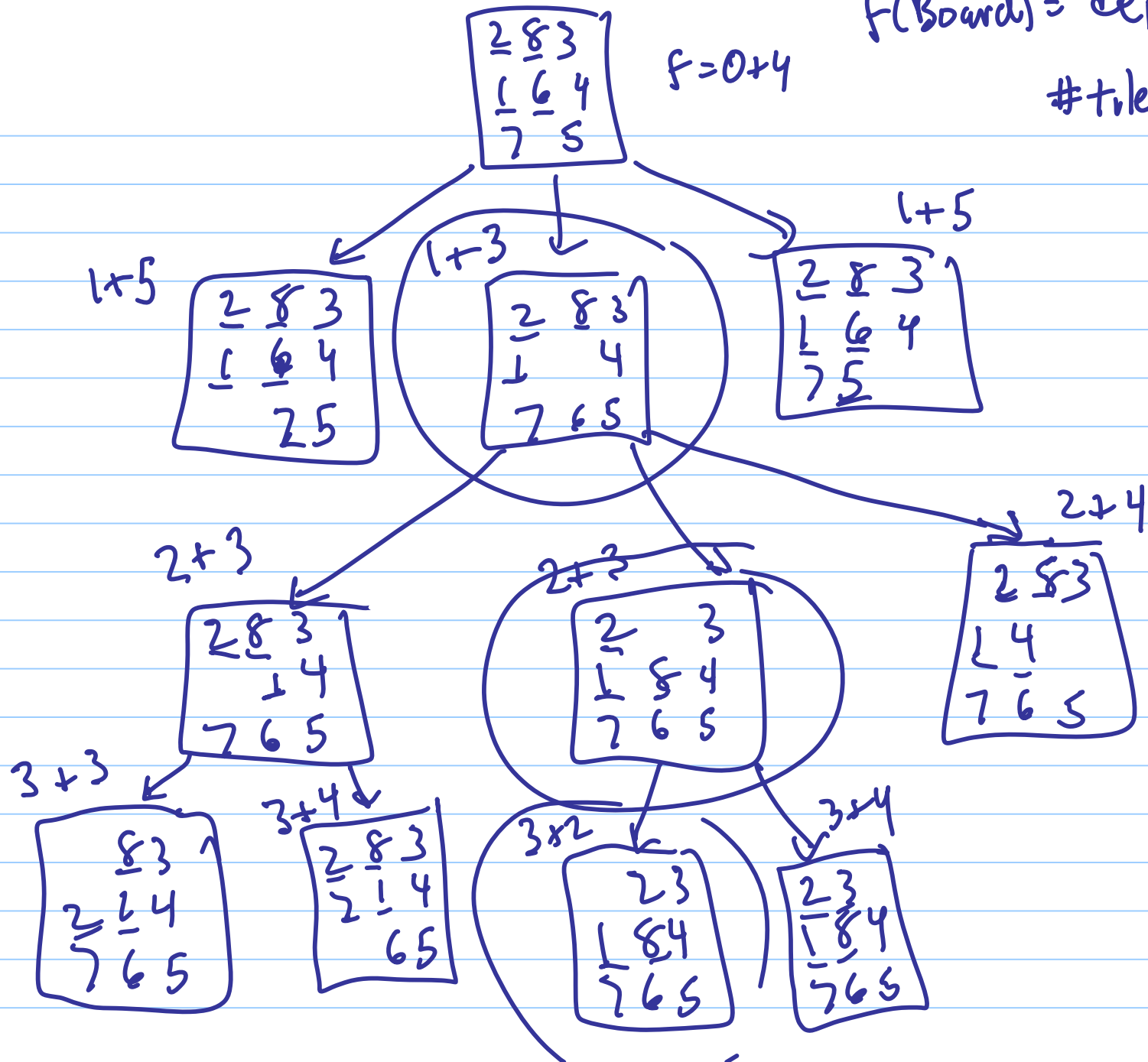
2	8	3
1	6	4
7		5

$$f() = 4$$

goal

1	2	3
8	4	
7	6	5

$$f(\text{Board}) = \text{depth in tree} + \text{\# tiles out of place}$$



Programming w/ Backtrack to solve certain AI problems

- how to represent a state

how do you represent the knowledge needed for this problem

Missionaries + Cannibals:

(a) list: left: MMCCB right: MC

(b) 2 2 L 1 1

1
M on left C on left boat pos 1 M on right, etc.

function AllPossibleMoves (state) :

for each m in ALL POSSIBLE ACTIONS

if precondition (m, state)

possibleMoves . Add (m)

return possibleMoves

function goal (state) :

// for M + C problem -

is state == "00;33 Boat"

Apply Move:

(+2 0 R)
(-2 0 L)

// for m+c problem:

if $(m, n, B, x, y) = \text{state}$

$\underbrace{\quad\quad\quad}_{\# \text{ on left}}$ \downarrow $\underbrace{\quad\quad\quad}_{\# \text{ on right}}$

boat
pos

move = (a, b, dir)

$\underbrace{\quad\quad\quad}_{\# \text{ to move}}$ \downarrow L or R

new state = either $(m+a, n+b, \text{dir}, x-a, y-b)$

or $(m-a, n-b, \text{dir}, x+a, y+b)$

Encoding for Moves:

$$\left[\begin{array}{l} \text{ALL MOVES} = (2\ 0)\ (1\ 0)\ (1\ 1)\ (0\ 1)\ (0\ 2) \\ \quad\quad\quad (-2\ 0)\ (-1\ 0)\ (-1\ -1)\ (0\ -1)\ (0\ -2) \end{array} \right]$$

a move (x, y) means

move x missionaries + y cannibals from
left bank to right bank

$$\text{STATE} = (3\ 3\ L)$$

$$\text{new STATE} = (3+x, 3+y, R)$$

Write a function
Apply Move (state, move)
That returns the
new state after
applying move.

Initial State = (3 3)

ALL MOVES = (2 0) (1 0) (1 1) (0 1) (0 2)
(-2 0) (-1 0) (-1 -1) (0 -1) (0 -2)

Precondition (state, move) = TRUE if : $0 \leq m+x \leq 2$ &&
 $0 \leq c+y \leq 2$ &&
 $(m < pos)$ (x, y)

position of boat is appropriate:

pos = RIGHT $\Rightarrow x, y \geq 0$

pos = LEFT $\Rightarrow x, y \leq 0$

Apply Move (state, move)

Returns (m+x, c+y, other pos)

goal : state == (0 0)

deadend { Cannibals Kill
Missionaries
(Film at 11) }

(mL > 0 && cL > mL) ||
(mR > 0 && cR > mR)

given (m, c) = state
m : # Missionaries on Left = mL
c : # Cannibals on Left = cL
3-m : # Missionaries on Right = mR
3-c : # Cannibals on Right = cR

