# Tentative Search Strategies

Note Title

Simple search strategies

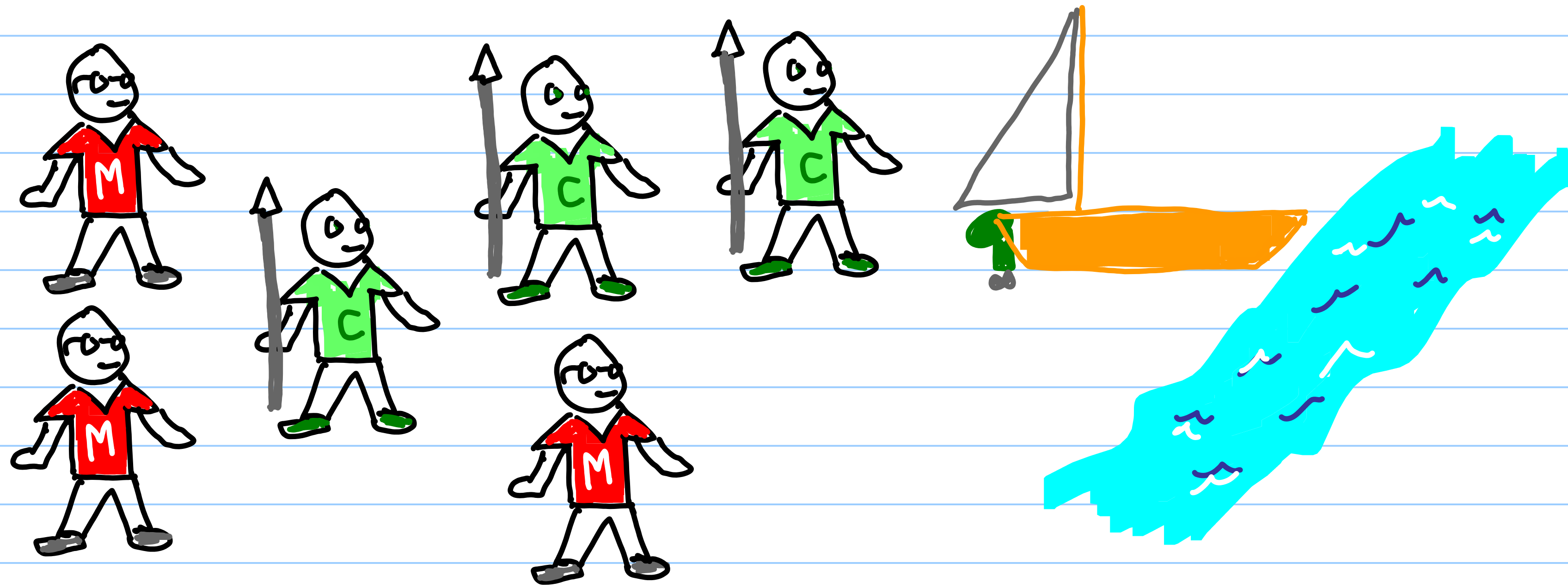
- "flail"
- "hill climbing" — need quality function that improves as state improves
  — quit when reaching max — (local max)

<u>adaptations</u> :

- try multiple starting points
- relax the "must improve on each move" restriction — (e.g. Simulated Annealing, Tabu Search)
- may stop at a local optimum which is not global optimum

**Missionaries and Cannibals:**

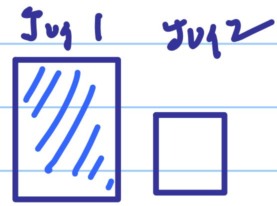
Three missionaries and three cannibals come to a river. There is a boat on their side of the river that can be used by either one or two persons. How should they use this boat to cross the river in such a way that cannibals never outnumber missionaries on either side of the river? (It is permissible for one or more cannibals to be alone on a bank that has no missionaries on it).



# Backtracking

Basic idea — can try something tentatively - if that doesn't work, can backtrack + try something else —

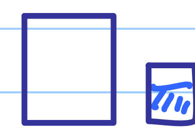
## Water Jugs problem:




| | | |
|---|---|---|
| capacity | 5 | 2 |
| amount | 5 | 0 |

state: (5 0)

problem — Find a sequence of legal moves yielding the following state:



| | | |
|---|---|---|
| capacity | 5 | 2 |
| amount | 0 | 1 |

state: (0 1)

rules: can pour everything out of a jug

state: $(x \quad y)$

| | Rule A | Rule B | |
|---|---|---|---|
| | $x > 0$ | $y > 0$ | ← precond |
| rule: | $(-x \quad 0)$ | $(0 \quad -y)$ | ← action |
| apply rule: | $(0 \quad y)$ | $(x \quad 0)$ | ← result |

Rule A     Rule B

another type of rule:

pour as much from Jug 1 into Jug 2 as possible

$amt_1 \quad amt_2 \quad cap_1 \quad cap_2$    state

Rule $_{1 \to 2}$ $\min\{amt_1, \; cap_2 - amt_2\}$ →

Rule $2 \to 1$ $\min\{amt_2, \; Cap_1 - amt_1\}$

| state | | rule | result |
|---|---|---|---|
| $(5 \quad 0)$ | ⇒ | $\min(5, 2) = 2$ | $(-2, 2)$ |
| $(5 \quad 1)$ | ⇒ | $\min(5, 1) = 1$ | $(-1, 1)$ |
| $(1 \quad 0)$ | ⇒ | $\min(1, 2) = 1$ | $(-1, 1)$ |
| $(1 \quad 1)$ | ⇒ | $\min(1, 1) = 1$ | $(-1, 1)$ |

↑ state    ↑ rule    ↑ result

$Rule_{1\to 2}$ :

pre cond

$$\boxed{\begin{array}{l} amt_1 > 0 \\ amt_2 < cap_2 \end{array}}$$

← i.e, Jug 1 non-empty
&
← Jug 2 not full

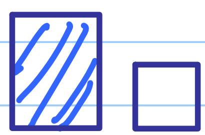
action

State
$(x, y)$
$amt_1$ $amt_2$

$$\boxed{\begin{array}{l} (x,y) \to \\ (\,x - \min(amt_1,\ cap_2 - amt_2) = x - \min(x,\ cap_2 - y) \\ \ \ y + \min(amt_1,\ cap_2 - amt_2) = y + \min(x,\ cap_2 - y) \end{array}}$$

i.e,

$$\boxed{\begin{array}{l} \text{let } q = \min(amt_1,\ cap_2 - amt_2) \\ (x,y) \to (\,x - q,\ y + q) \end{array}}$$

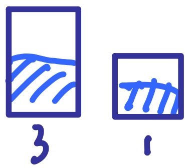


5 0

$q = \min(5, 2)$

$(x, y) \to (\,x - 2,\ y + 2) = (3, 2)$

$(5, 0)$




$y = \min(3, 2-1) = 1$

$(x,y) \to (3-1, 1+1) = (2,2)$

$\;\;\,\| $

$(3,1)$

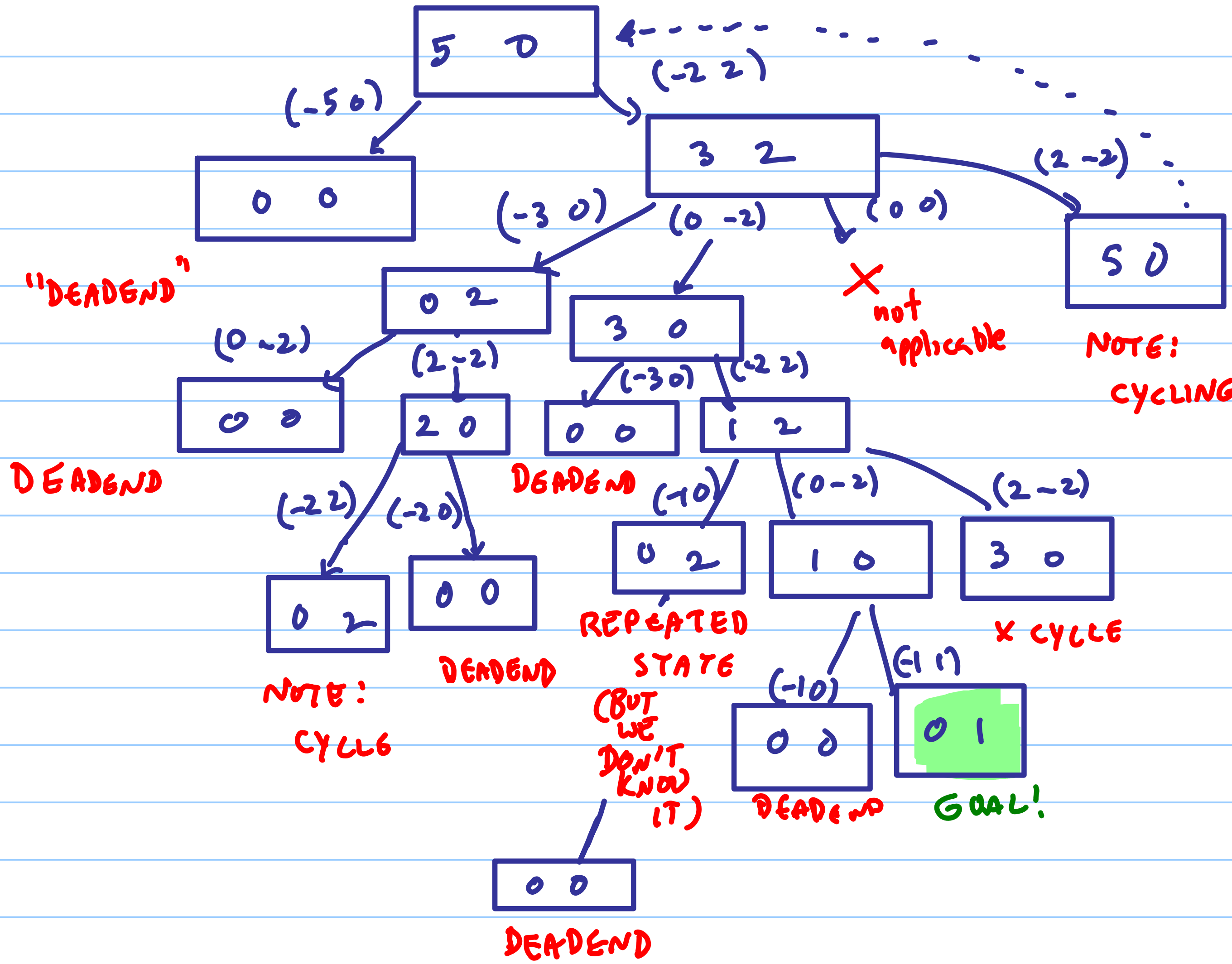
---

Initial state — $(5 \;\; 0)$

Applicable Rules

| | | |
|---|---|---|
| | $(-5 \;\; 0)$ | pour all out of Jug 1 |
| precond false X | $(0 \;\; 0)$ | pour all out of Jug 2 |
| | $(-2 \;\; 2)$ | pour from Jug 1 to Jug 2 |
| precond false X | $(0 \;\; 0)$ | pour from Jug 2 to Jug 1 |

# Completed Search Tree



Backtrack ( State List )

Path from Start to current state

Backtrack Version Ø

returns a path from start to goal or 'FAILED'

... sequence of rules from start to goal

```
state = first ( Statelist )
if goal ( state ) return NULL
if state ∈ rest ( Statelist ) return "FAILED"   ( reason : CYCLE )
if deadend ( state ) return "FAILED"
MOVES = ApplicableRules ( state )
for each m ∈ MOVES
    nextState ← applyRule ( m, state )
    newStateList ← nextState + Statelist   (add nextState to front of Statelist)
    path = Backtrack ( newStateList )
    if Path ≠ "FAILED"
        return m + path
return "FAILED"   (reason : no more worked from current state)
```