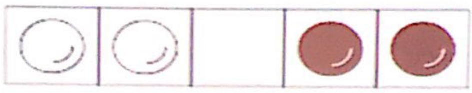
Solution

Well-defined formulation of the stone puzzle:



States:

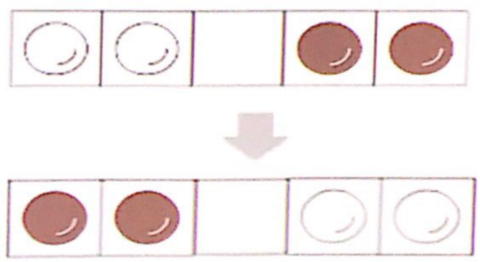
square content - 5 variables, 3 values each:
white (O), black (X), empty (–)

(– X X O O)

position of each stone? *not* good

Initial state:

(O O – X X)



Goal test:

state equal to (X X – O O)

Solution

Well-defined formulation of the stone puzzle:

Operators:

To define every step? (O O – X X) →(O – O X X)
=> too many! Need to be abstract:

- MoveToRight: (O–) → (–O)
- MoveToLeft: (–X) →(X–)
- JumpToRight: (OX–) → (–XO)
- JumpToLeft: (–OX) → (XO–)

Path cost:
Number of operators used (1 for all ops)

Solution

Problem search tree and solution:

To define a search tree:

- valid, reachable states only (subset of the state space)
- symmetric portion of the search tree not shown

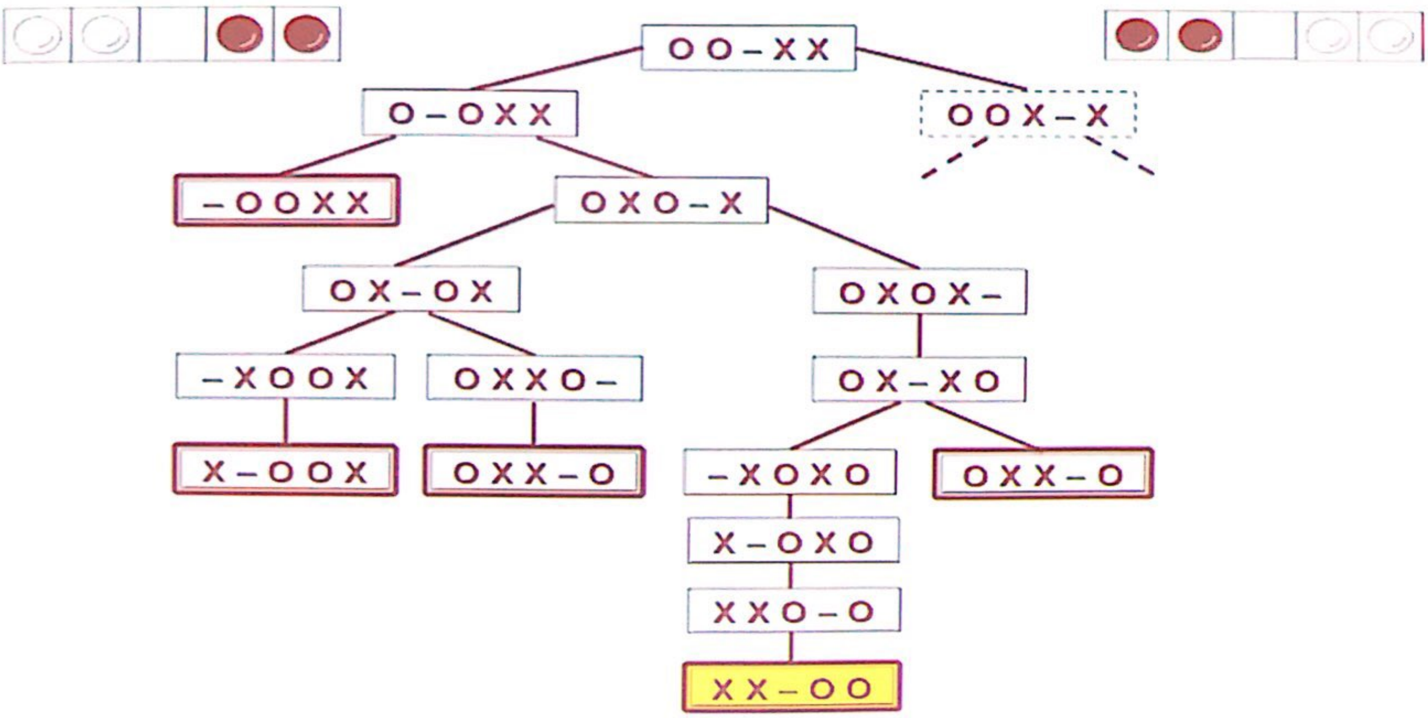
Initial state: (O O – X X)

Operators:

- MR: (O –) → (– O)
- ML: (– X) → (X –)
- JR: (O X –) → (– X O)
- JL: (– O X) → (X O –)

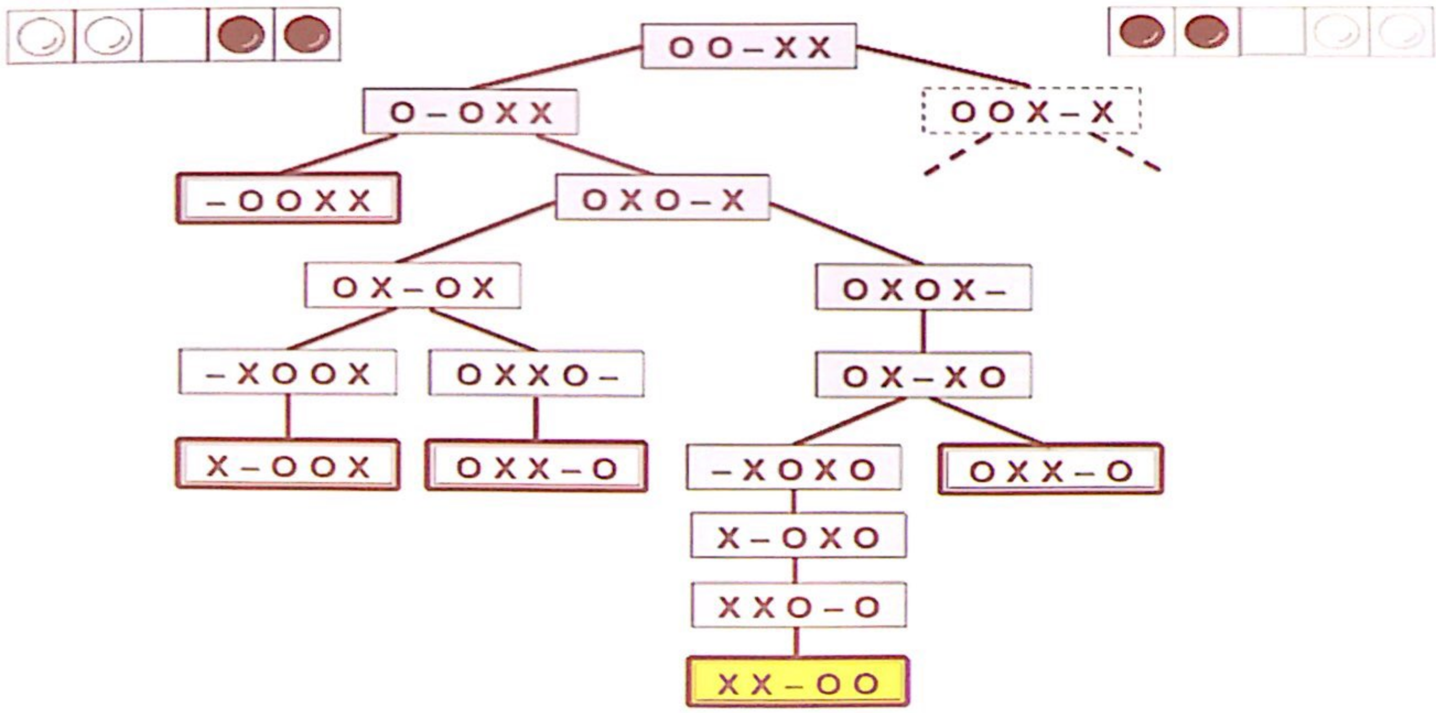
Solution

Problem search tree and solution:



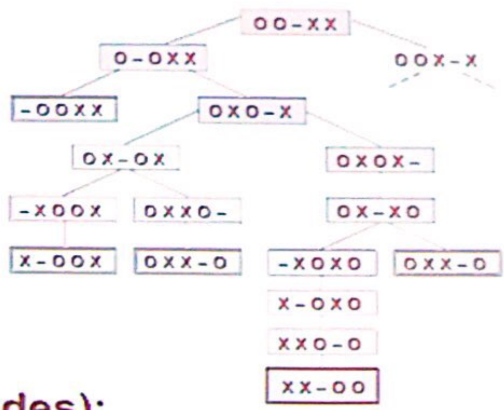
Solution

Problem search tree and solution:



Solution

Characteristics of the search space:



- **Number of branches:**
  $15 \times 2 = 30$
- **Non-terminal nodes:**
  $1 + 2 * 10 = 21$
- **Average branching factor:**
  $30 / 21 \approx 1.43$
- **Depth of the 2 solutions:**
  8
- **Space complexity (memory to store the nodes):**
  - Actual space required = 31 nodes
  - Theoretical space required = $1 + 1.43 + 1.43^2 + \ldots + 1.43^8 \approx 55$

Note: "average" is not very relevant if search space is small, based on a *uniform* search tree; here d=8 for solution path and only 5 otherwise!

Solution

Most suitable search algorithm:

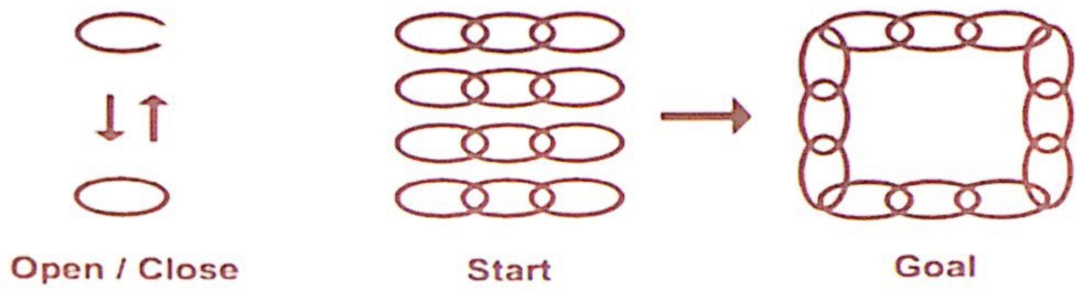
- **Heuristic function?**
  - No → uninformed search (else A*)
- **Optimal solution?**
  - low branching factor & equal cost → BFS, (else IDS)
  - variable operator cost? → UCS
- **Any solution ok?**
  - DFS (else IDS)

Note: for small problems, *any* algorithm will do!

Q2.2

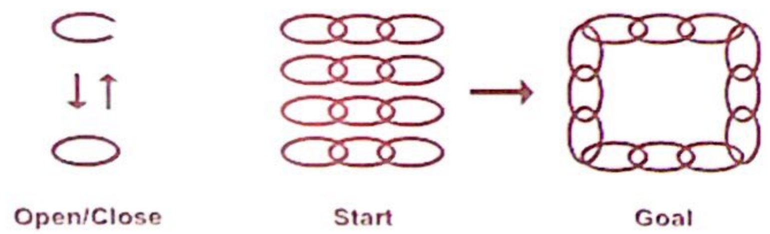
2.2 The chain problem consists of re-arranging a set of chains of various lengths into another set as required. In the example shown below, the initial set comprises four non-circular chains of three links each, while the final set consists of a single circular chain of twelve links. The two possible operations consist of opening a closed link and closing an open link, respectively, as illustrated. Open links can be added to a non-circular chain at either end or else join both ends into a circular chain. Closed links can be removed without restriction.

Provide a suitable definition of *states* and *operators* for the chain problem, which could be used by a problem-solving agent. Show a sequence of operators yielding a possible solution (without using a search algorithm…)



Solution

Formulation of the chain problem:



States:

- set of *n* chains (start n=4, goal n =1)
- chains of *k* links, circular or not (*l* = 0 or 1)
- links open or closed (open: *c* = 1 or closed: c = 0)

  → { ... (*k*, *l*, *c*) ... } (note: *c*=0 for *k*>1)

Initial state:

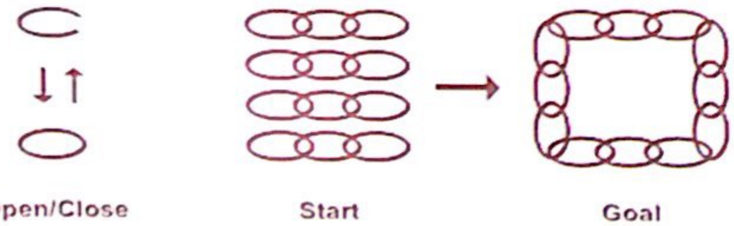
{ (3,0,0) (3,0,0) (3,0,0) (3,0,0) }

Goal state:

{ (12,1,0) }

Solution

Formulation of the chain problem:



Operators: *"open"*

- OS: *open a single link* (1,0,0) → (1,0,1)
- OE: *open a link at the end of a chain*

  (k,l,0) → (1,0,1) + (k–1,0,0) for $k > 1$
- OM(m): *open a link in the middle of a chain*

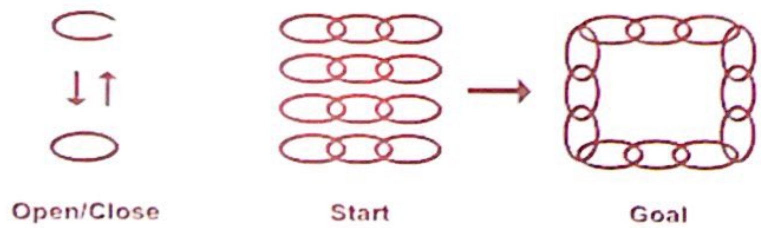
  (k,0,0) → (1,0,1) + (m,0,0) for k > 2

  \+ (k–m–1,0,0) for k–1>m>0

  ✓ Example: {(3,0,0)} → {(1,0,0),(1,0,1),(1,0,0)}

Solution

Formulation of the chain problem:



Operators: *"close"*

- CS: *close a single link* (1,0,1) → (1,0,0)
- CE: *close a link at the end of a chain*

  (1,0,1) + (k,0,0) → (k+1,l,0)
- CM: *close a link in between two chains*

  (k,0,0) + (m,0,0) + (1,0,1) → (k+m+1,0,0)

note: symbols can be abstracted to O() and C() only

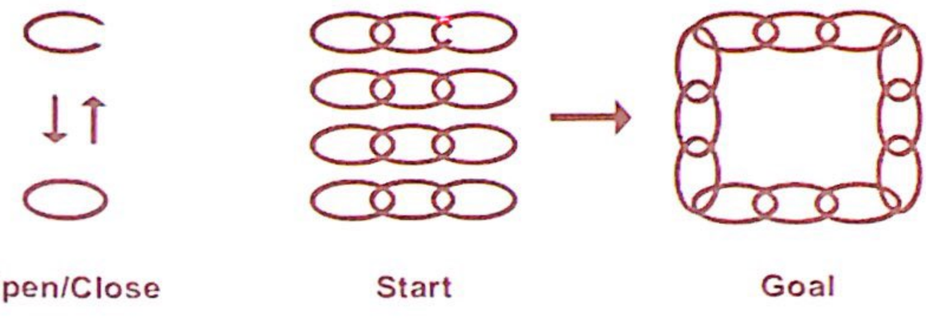
Path cost:

- Number of operators applied (1 for all ops)

Solution

Optimal solution to the chain problem:

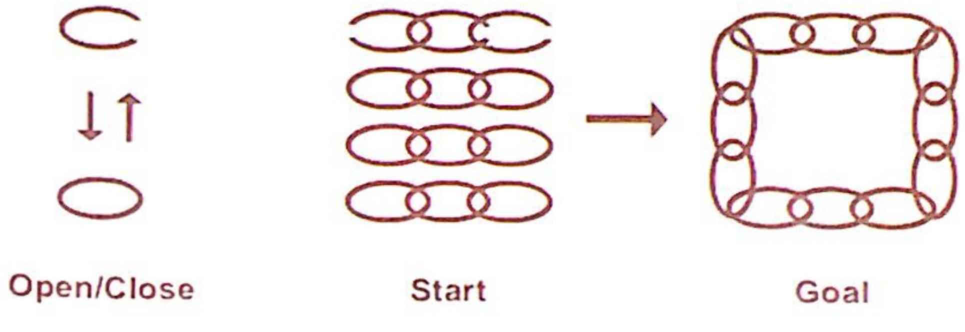
- **Initial:** { (3,0,0), (3,0,0), (3,0,0), (3,0,0) }
- **OM(1):** { (3,0,0), (3,0,0), (3,0,0), (1,0,1), (1,0,0), (1,0,0) }



Solution

Optimal solution to the chain problem:

- **Initial:** { (3,0,0), (3,0,0), (3,0,0), (3,0,0) }
- **OM(1):** { (3,0,0), (3,0,0), (3,0,0), (1,0,1), (1,0,0), (1,0,0) }
- **OS():** { (3,0,0), (3,0,0), (3,0,0), (1,0,1), (1,0,1), (1,0,0) }
- **OS():** { (3,0,0), (3,0,0), (3,0,0), (1,0,1), (1,0,1), (1,0,1) }



Solution

Optimal solution to the chain problem:

- **Initial:** { (3,0,0), (3,0,0), (3,0,0), (3,0,0) }
- **OM(1):** { (3,0,0), (3,0,0), (3,0,0), (1,0,1), (1,0,0), (1,0,0) }
- **OS():** { (3,0,0), (3,0,0), (3,0,0), (1,0,1), (1,0,1), (1,0,0) }
- **OS():** { (3,0,0), (3,0,0), (3,0,0), (1,0,1), (1,0,1), (1,0,1) }
- **CM():** { (7,0,0), (3,0,0), (1,0,1), (1,0,1) }
- **CM():** { (11,0,0), (1,0,1) }
- **CE(1):** { (12,1,0) }

6 steps only