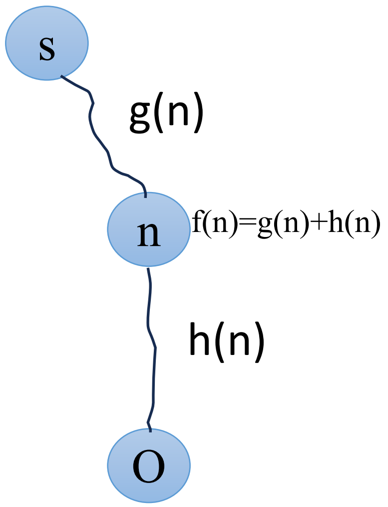
AIFA
BEST FIRST SEARCH

16/01/2024

Koustav Rudra

BEST-FIRST Tree Search



- Initialize: Set OPEN={s}, CLOSED = {}, f(s) = h(s)

- Fail:
  - If OPEN={}, Terminate with failure

- Select: Select the minimum cost state, n, from OPEN and save in CLOSED

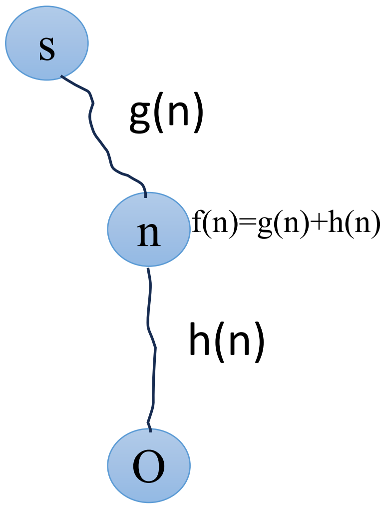
- Terminate:
  - If n∈G, terminate with success

BEST-FIRST Tree Search

BEST-FIRST Tree Search with pruning

- Expand:
  - For each successor, m, of n:
    - If m∉[OPEN∪CLOSED]
      - Set $f(m) = h(m)$
      - Insert m in OPEN
    - If m∈[OPEN∪CLOSED]
      - Set $f(m) = h(m)$
      - If f(m) has decreased and $m \in CLOSED$
        - Move m to OPEN
- Loop:
  - Go to step 2

BEST-FIRST Tree Search [pruning]



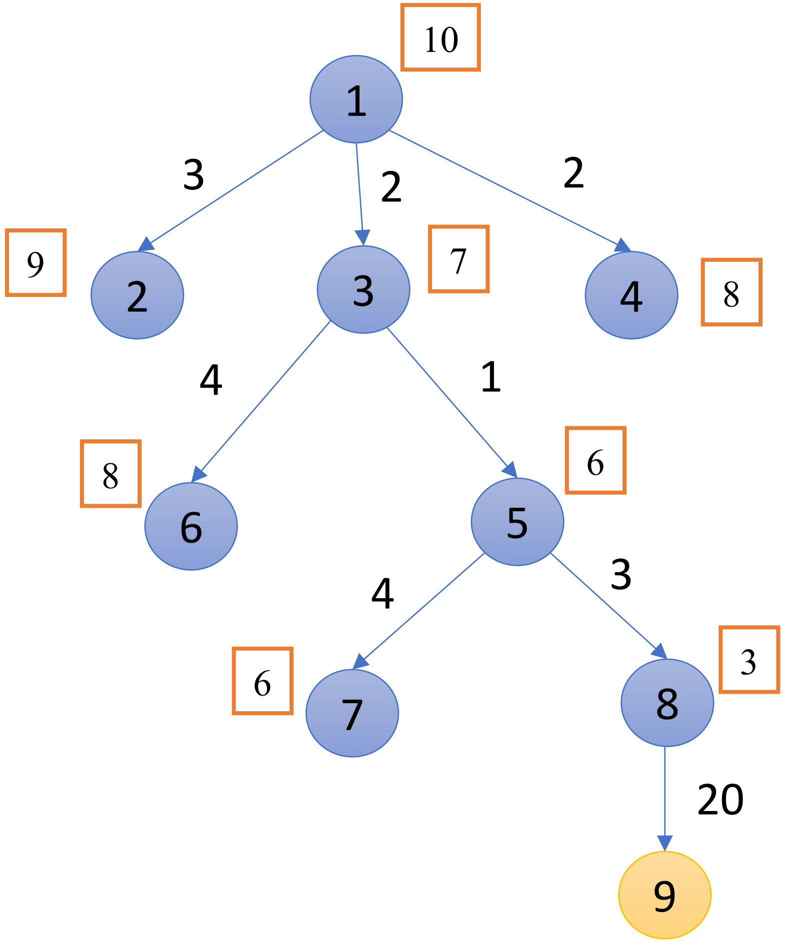
- Initialize: Set OPEN={s}, CLOSED = {}, f(s) = h(s), CB
- Fail:
  - If OPEN={}, Terminate with failure
- Select: Select the minimum cost state, n, from OPEN and save in CLOSED
- Terminate:
  - If n∈G and f(n) < CB, CB = f(n), Go to Step 2
  - Else terminate

BEST-FIRST Tree Search [pruning]

- Expand:
  - If f(n) ≤ CB
    - For each successor, m, of n:
      - If m∉[OPEN∪CLOSED]
        - Set $f(m) = h(m)$
        - Insert m in OPEN
    - If m∈[OPEN∪CLOSED]
      - Set $f(m) = h(m)$
      - If f(m) has decreased and $m \in CLOSED$
        - Move m to OPEN

- Loop:
  - Go to step 2

BEST-FIRST Tree Search



OPEN SET	SELECT	GOAL	EXPANDED	CLOSED
[1(10)]	1(10)	N	[2(9),3(7),4(8)]	[1(10)]
[2(9),3(7),4(8)]	3(7)	N	[2(9),4(8),6(8),5(6)]	[1(10),3(7)]
[2(9),4(8),6(8),5(6)]	5(6)	N	[2(9),4(8),6(8),7(6),8(3)]	[1(10),3(7),5(6)]
[2(9),4(8),6(8),7(6),8(3)]	8(3)	N	[2(9),4(8),6(8),7(6),6(0)]	[1(10),3(7),5(6),8(3)]
[2(9),4(8),6(8),7(6),9(0)]	9(0)	Y		

AIFA
Hill Climbing Algorithm

16/01/2024

Koustav Rudra

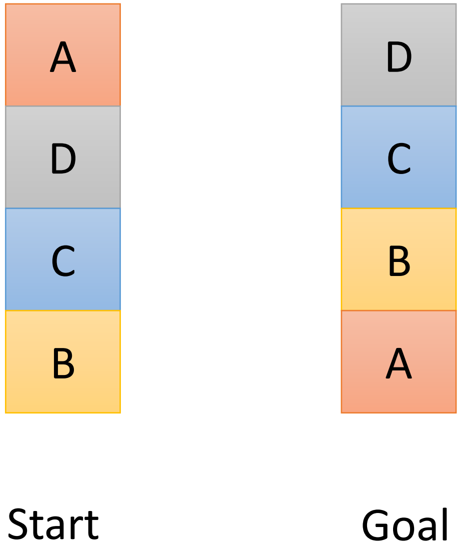
Hill Climbing Algorithm

- Evaluate the INITIAL state
  - If it is GOAL return it
  - Else CURRENT← INITIAL

- Loop until the solution is found or no new operators could be applied to CURRENT:
  - Select an operator that has not been applied to the current state [CURRENT] and apply it to produce new state [NEW]

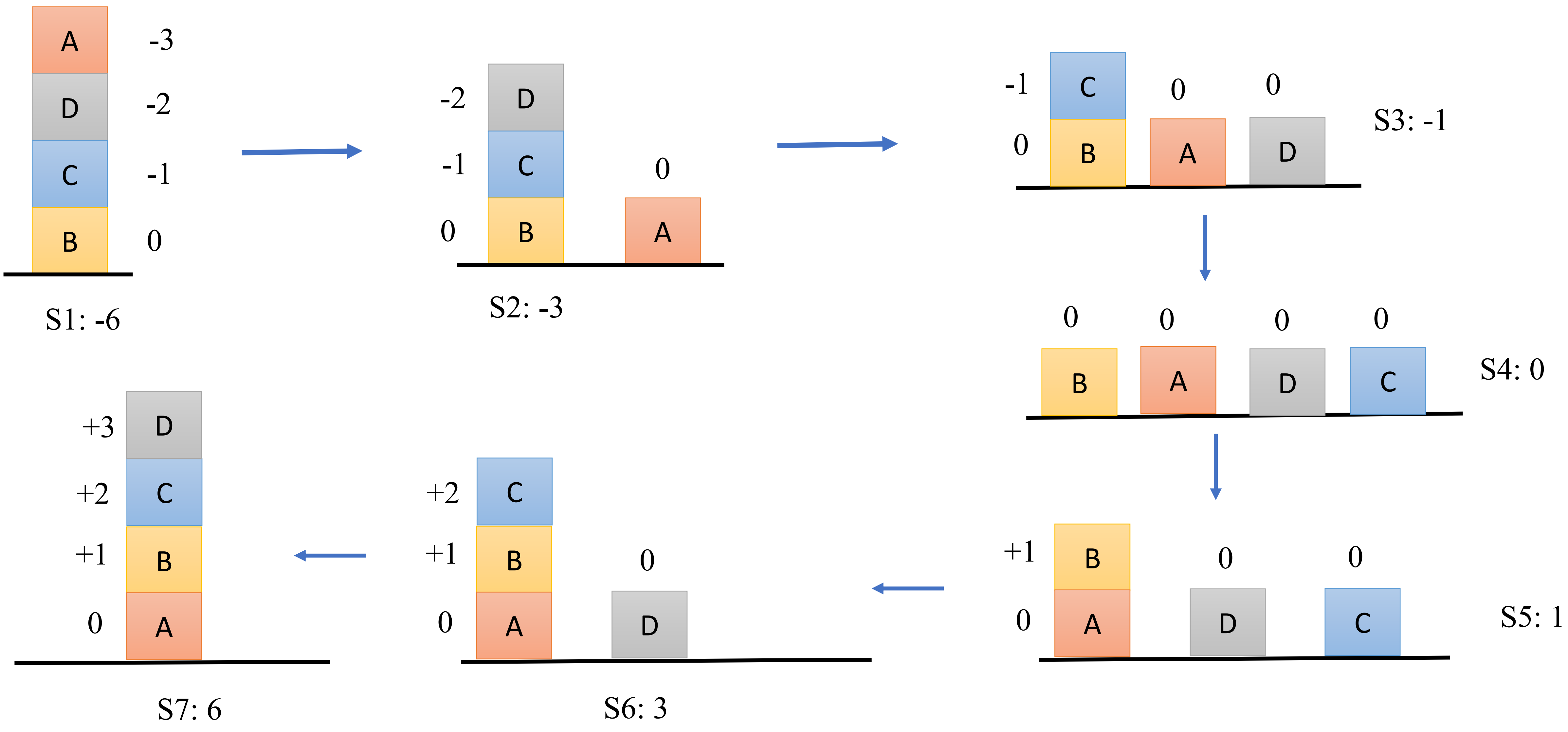
  - Evaluate NEW:
    - If it is GOAL return it
    - Else If NEW > CURRENT, CURRENT← NEW
    - Else go to Loop

Hill Climbing Algorithm



- h(x) = +1 for all the blocks in support structure if the block is positioned correctly
- Otherwise -1 for all the blocks

Hill Climbing Algorithm

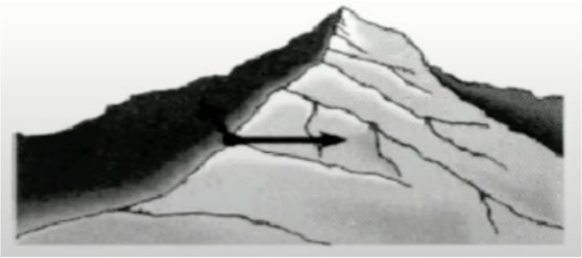


Hill Climbing Algorithm: Drawbacks

- Local Maxima:
  - A local maxima is supposed to be global maxima

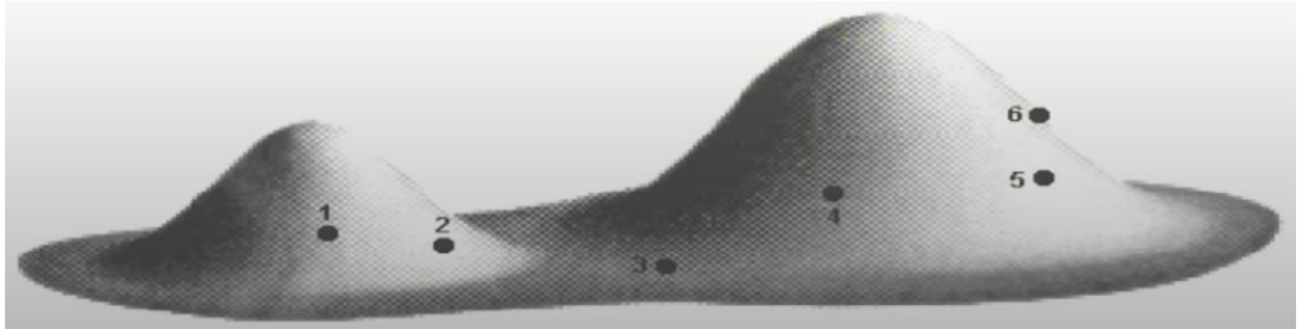
- Plateaus:
  - Area of sear h space where evaluation function is flat
  - Requiring random walk

- Ridge:
  - Steep slopes
  - Search direction is not towards the top but towards the side



Drawback: Solution

- In each of the previous cases (local maxima, plateaus, & ridge), the algorithm reaches a point at which no progress is being made



- Solution
  - Random-restart-hill-climbing
  - Random initial states are generated
  - Running each until it halts or makes no discernible progress
  - Best result is chosen

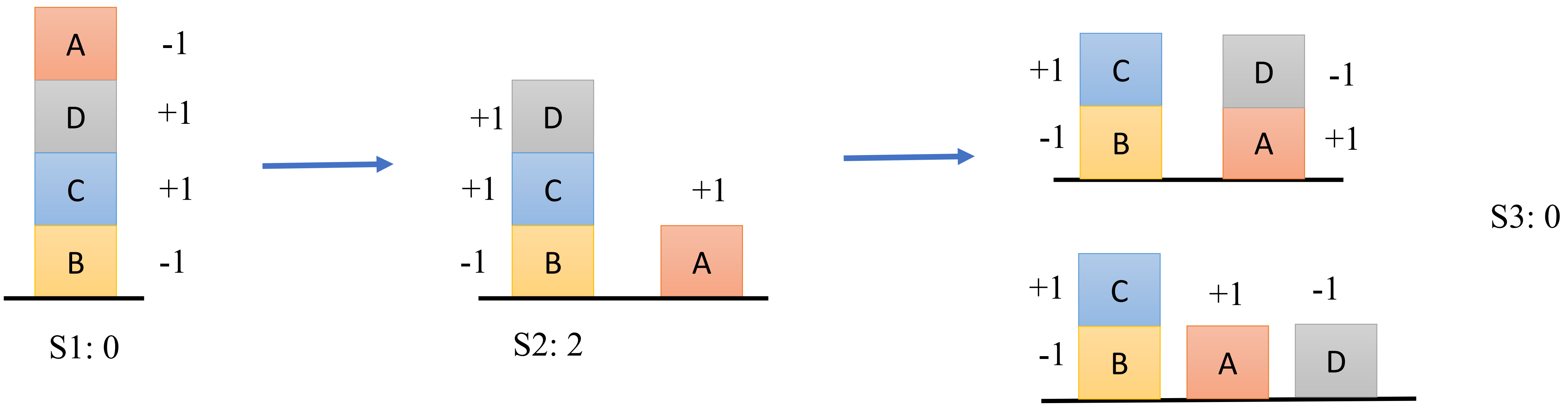
Hill Climbing Algorithm: Disadvantages

- Hill Climbing uses local information:
  - Decides what to do next by looking only at the “immediate” consequences of its choices
  - Will terminate when at local optimum
  - The order of application of operators can make a big difference
- Global information might be encoded in heuristic functions

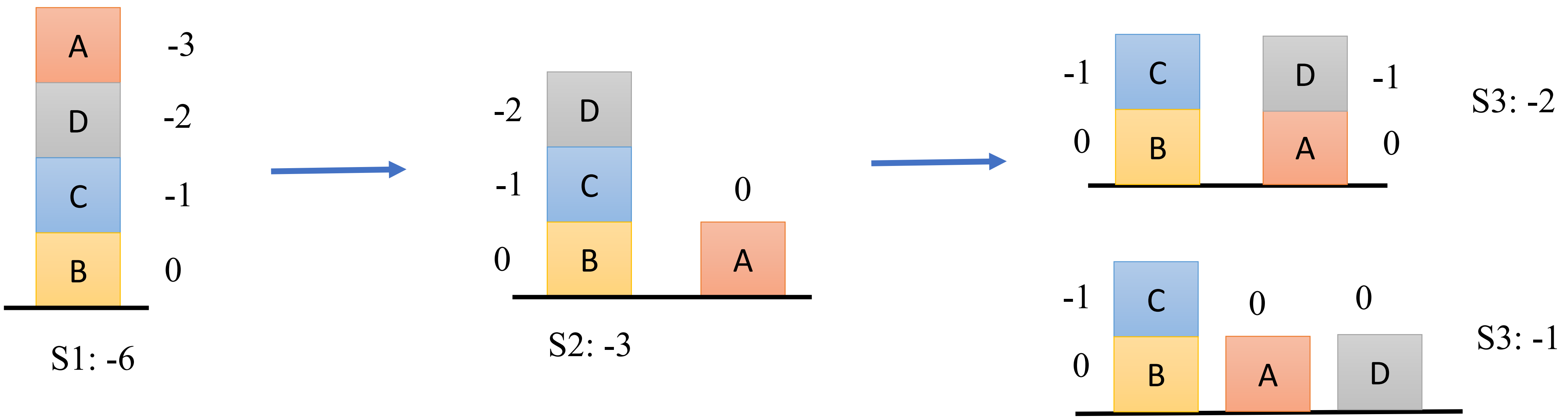
Hill Climbing Algorithm: Disadvantages

- Local Heuristics
  - +1 for each block that is resting on the thing it is supposed to be resting on
  - -1 for each block that is resting on a wrong thing

Hill Climbing Algorithm: Local Heuristics



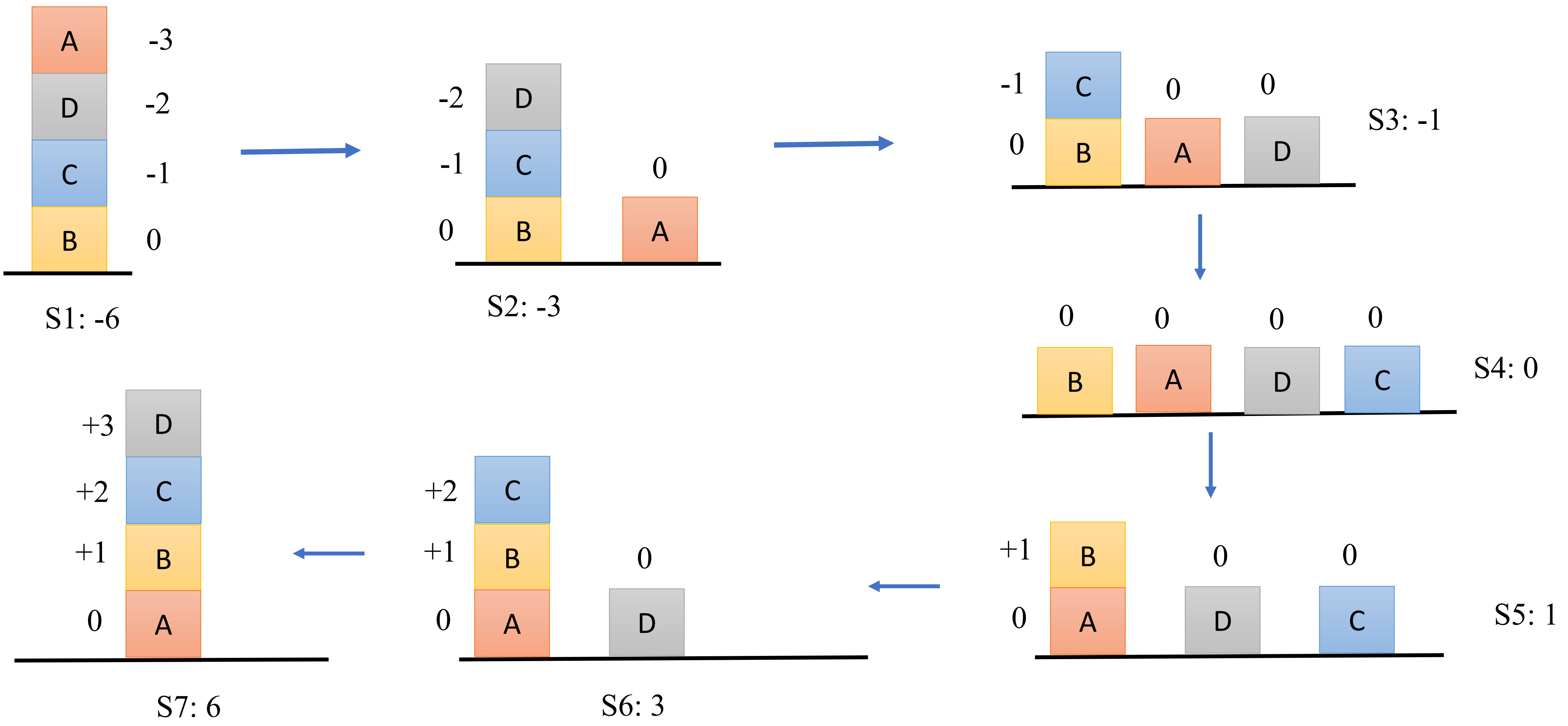
Hill Climbing Algorithm: Global Heuristics



- h(x) = +1 for all the blocks in support structure if the block is positioned correctly
- Otherwise -1 for all the blocks

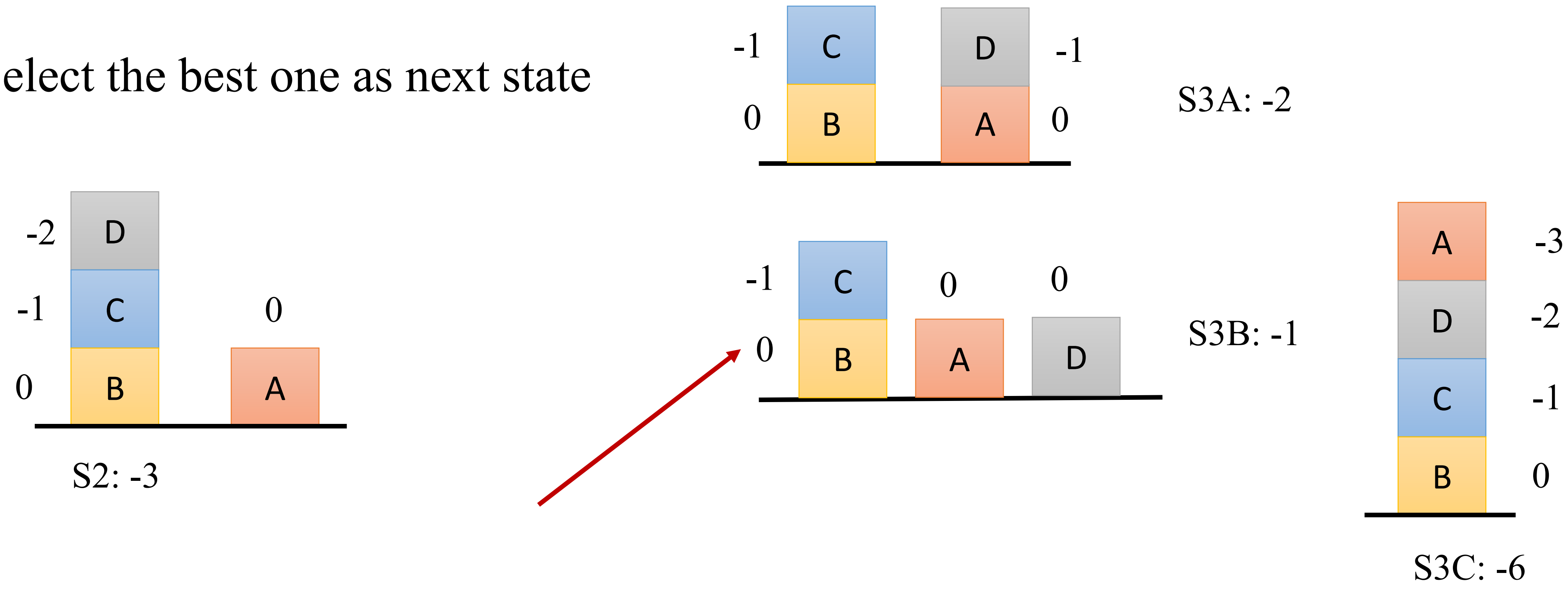
- There is no local maximum
- Takeaway
  - Sometimes changing the heuristic function is all we need

Hill Climbing Algorithm: Global Heuristics



Steepest-Ascent Hill Climbing Algorithm

- Basic Hill Climbing first applies one operator and gets new state
- Steepest-Ascent Hill Climbing considers all the moves from the current state
- Select the best one as next state



Steepest-Ascent Hill Climbing Algorithm

- Evaluate the INITIAL state
  - If it is GOAL return it
  - Else CURRENT$\leftarrow$ INITIAL

- Loop until the solution is found or until a complete iteration produces no change to CURRENT:
  - Let SUCC be a state such that any possible successor of the current state will be better than SUCC
  - For each operator that applies to the current state [CURRENT] do
    - Apply the operator and generate a new state [NEW]

    - Evaluate NEW:
      - If it is GOAL return it and quit
      - Else If NEW > SUCC, SUCC$\leftarrow$ NEW
  - If SUCC>CURRENT
    - CURRENT $\leftarrow$ SUCC

Search with Limited Memory and Time

- Mechanisms for discarding nodes
  - Pruning
  - DFBB
  - Memory Bound A*

- Mechanisms for redoing nodes
  - IDA*

- Domain relaxation – What is the utility?

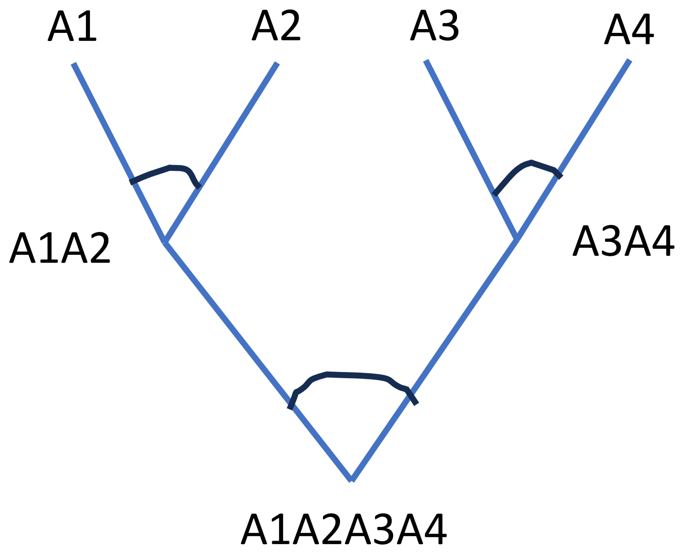
AIFA
Problem Reduction Search

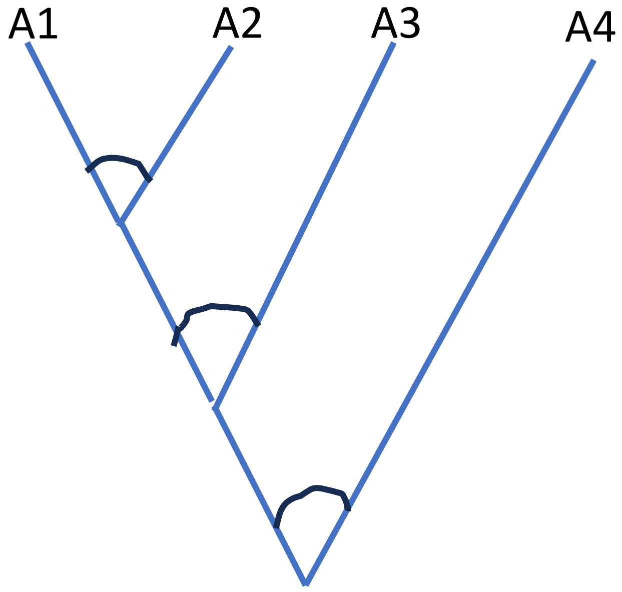
16/01/2024

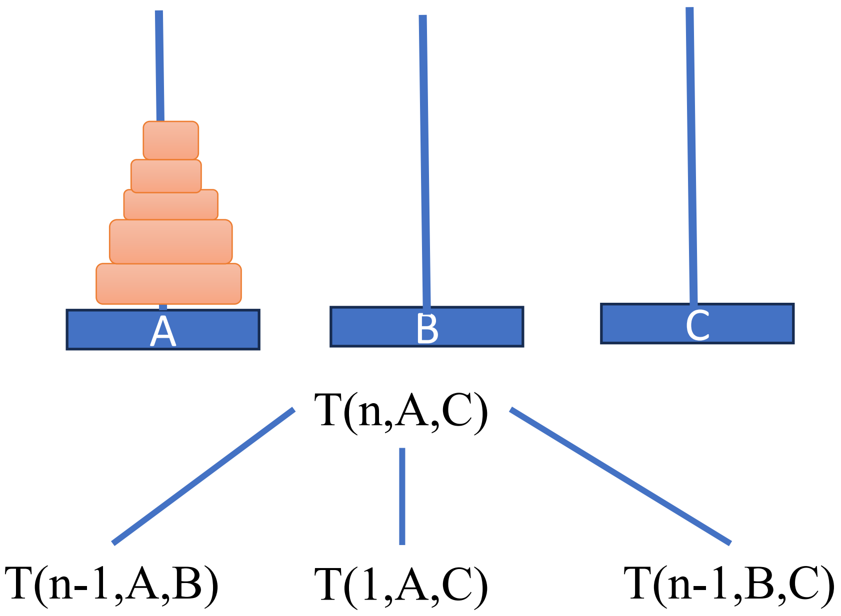
Koustav Rudra

Problem Reduction Search

- Planning how best to solve a problem that can be recursively decomposed into sub-problems in multiple ways
  - Matrix multiplication problem
  - Tower of Hanoi
  - Theorem proving





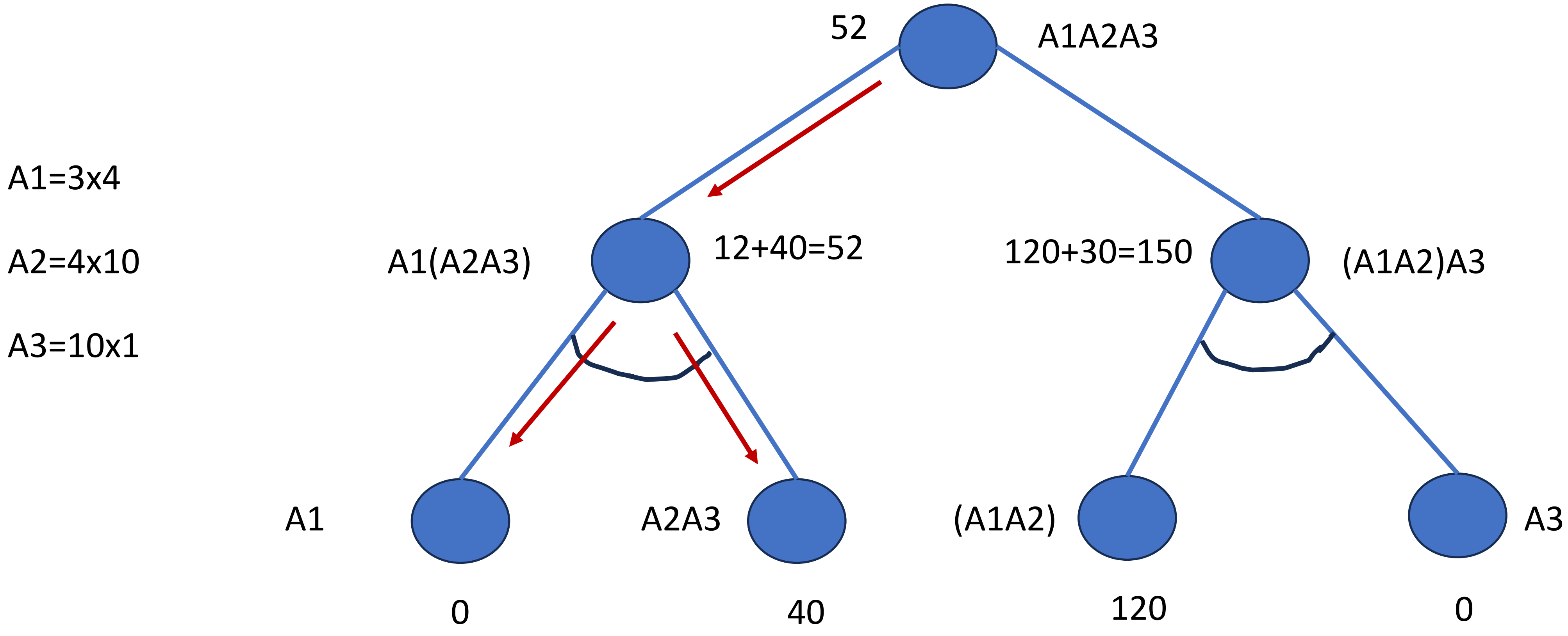


Formulation

- AND/OR Graph
  - An OR node represents a choice between possible decompositions
  - An AND node represents a given decomposition

- Game Trees
  - Max/Min nodes
  - Max nodes represent the choice of my opponent
  - Min nodes represent my choice

Each node has a separate optimization criteria



- This is when heuristics is not present

Thank You