# Section 1 – Dynamic Programming Pseudocode

Setup/Initialise cache

Function Recursive\_Search(State)

Best payoff = payoff(state)

Best action list = []

Best final state = state

For each child action in all valid actions

Child state = result(state, child action)

Child payoff, child action list, child final state = Recursive\_Search(child state)

If child payoff is better than best payoff

Best payoff = child payoff

Best action list = child action + child action list

Best final state = child final state

Return best payoff, best action list, best final state

# Section 2 – Branch and Bound Pseudocode

Branch\_Bound(mine)

Node = initial mine node

Frontier = Priority que of nodes ordered by payoff

Frontier += node

Best node = node

While frontier

Node = next node in frontier

For child node in all child nodes

If child node is not in frontier

Add child node to frontier

Else

If child node payoff is better than current payoff

delete incumbent node

best node = child node

add child node to frontier

return best payoff, best action list, best final state

# Section 3 – Testing Methodology

# Section 4 – Performance and Limitations