

# COMP9016 Assignment 1

Name: Melwyn D Souza

Date: 24/10/2021

Student Number: R00209495

Email: melwyn.dsouza@mycit.ie

Module: Knowledge Representation

Lecturer: Dr Ruairi O'Reilly

## Building My World

Below is a gold mining environment with three agent types (Simple reflex, Model Based and Goal Based), these are implemented in the python script

### Gold Mining environment

This is a 2D grid environment which contains gold, pebbles, ticking sounds and explosives(bomb). The things in the environment are distinguished using different colours as shown in table 1

Thing	Cell Colour	
Agent	Blue	
Pebble	Black	
Ticking sound	Grey	
Bomb	Red	

Table 1 – Colours of 'Things' in gold mine environment

The 7x7 2D grid below (fig 1) with 10 golds, 4 Pebbles and 4 bombs

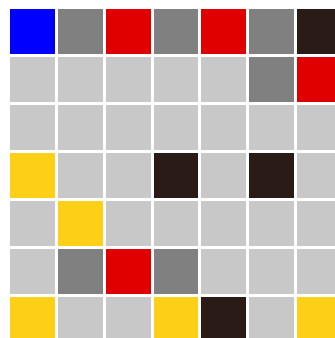


Fig 1 – 2D gold mining environment with golds, pebbles, ticking sounds and bombs

**Task environment** - the problem is formulated as follows,

- Initial state of agent can be any location, but I have assigned it to be [0,0] for this example
- Actions – Move forward if no things are found at the current location, turn left or right if bump is detected, collect things if things are detected at the current location
- Initially agent gets 20 points and 3 lives to start, the costs of agent's actions are mentioned in table 2
- The agent wins and game ends if he satisfies at least one condition below

- Points  $\geq 50$
- The agent collects all golds
- The game ends with a loss if agent is dead, this happens when agent loses all lives (When points  $\leq 0$ , the agent uses 1 life (-1 life) in exchange of +20 points)

Action	Cost (Points)	Cost (Lives)
One step	-1	0
Gold	+12	0
Pebble	-4	0
Bomb	-5	-1
Locate Bomb	+5	0

*Table 2 – Costs of agent's action in gold mining environment*

#### PEAS description

Agent	Performance Measure	Environment	Actuators	Sensors
Gold Digger	Collect enough gold to get 50+ points and win the game (Table 2)	2D 7X7 grid with golds, pebbles Ticking and bombs	Locate bomb, move forward, turn, left, right, collect	Bump (Wall), Ticking Sound

*Table 3 – PEAS description of gold mine environment*

## Simple Reflex Agent Type

A simple reflex is a dumb agent, it only responds to the current percept, it does not depend on the percept history. The condition action rule is carried out in the agent program, where an action is carried out when some condition is fulfilled

The dumb gold digger can only move in a zig zag pattern, it starts from location 0,0 and moves until a bump is sensed, it moves to next row and changes direction from left to right or right to left depending on its current location

The rules of the world are as described in PEAS, each step will cost -1 points, the agent will die once lives are 0, when points are 0 agent will -1 its lives and +20 the points to survive

Below is the simple reflex agent shown in fig 2, the white line on the grid shows the path travelled by the agent, as can be seen that the agent does not care about anything other than the current precept, it collects 3 bombs on its way and loses 3 lives (the game ends at second row) , table 4 shows few important steps and actions which cost over agent its lives

Cons:

- The example below depicts a drawback where the agent loses all its lives (3 bombs costing 3 lives) by the time it reaches 2<sup>nd</sup> row
- When the grid is very large, the agent has only 20 points and each step costs -1 point, the agent will exhaust all its points and lives since it does not check the shortest path for golds

Pros:

- When the things added to the environment are random, this agent could get lucky and find shortest distance faster than other agents since it only moves forward from one end to other

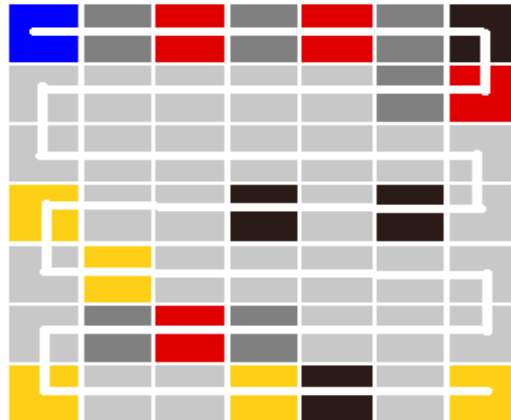


Fig 2 – Simple reflex gold digger agent (White line - path traversed)

<p>This is 1st step  Current location of agent is [0, 0]  Things at this location  [&lt;SimpleReflexGoldDigger&gt;]  Points: 19, Lives: 3  Moving forward</p>	
<p>This is 3rd step  Current location of agent is [2, 0]  Things at this location [&lt;Gold&gt;, &lt;Bomb&gt;,  &lt;SimpleReflexGoldDigger&gt;]  Points: 17, Lives: 3  My points are 29 and lives are 3  SimpleReflexGoldDigger collected Gold at  location: [2, 0]</p>	
<p>This is 7th step  Current location of agent is [4, 0]  Things at this location [&lt;Bomb&gt;,  &lt;SimpleReflexGoldDigger&gt;]  Points: 20, Lives: 2  My points are 15 and lives are 1  SimpleReflexGoldDigger collected Bomb at  location: [4, 0]</p>	

<p>This is 14th step</p> <p>Current location of agent is [6, 1]</p> <p>Things at this location [&lt;Bomb&gt;, &lt;SimpleReflexGoldDigger&gt;]</p> <p>Points: 28, Lives: 1</p> <p>SimpleReflexGoldDigger collected Bomb at location: [6, 1]</p> <p>Gold digger ran out of lives, agent is dead</p> <p>My points are 23 and lives are 0</p>	
---	--

Table 4 – Handpicked steps and actions from simple reflex gold digger agent

## Model Based Reflex Agent Type

The model based reflex agents keeps some knowledge of the world in its percept history and the future actions will not only be dependent on the current percept but also its percept history, hence this agent type is smarter than simple reflex and makes more informed decisions

The performance measures are same as described in PEAS and table 2, but model-based gold digger agent senses the ticking sounds of the bomb and locates the bomb in order to save its lives, it then changes direction to downward and then to right or left depending on its previous direction

Fig 3 shows the path traversed by my model-based agent, it can be noticed that once the agent hears the ticking sounds of the bomb, it never continues in that direction

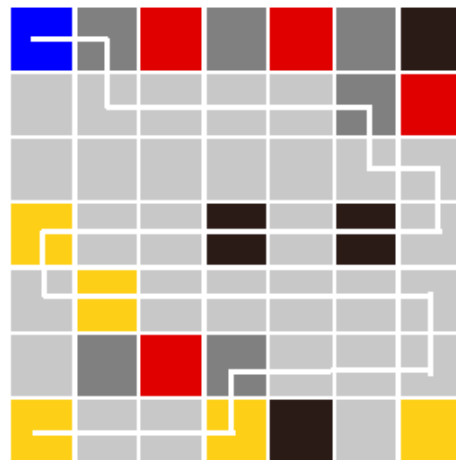


Fig 3 – Model based gold digger agent (white line - path traversed)

The table 5 lists few important steps carried out by our agent, we can see how the agent locates bomb and changes direction depending on percept history, also the agent wins (Points > 50) on 34<sup>th</sup> iteration with all lives assigned initially

<p>This is 2nd step</p> <p>Current location of agent is [1, 0]</p> <p>Things at this location [&lt;Tick&gt;, &lt;ModelBasedGoldDigger&gt;]</p> <p>Points: 18, Lives: 3</p> <p>There might be bomb around this location [1, 0]</p> <p>There are bombs in location [[2, 0]]</p>	
---	--

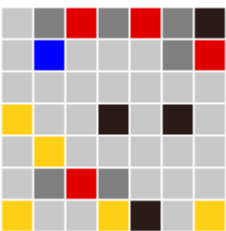
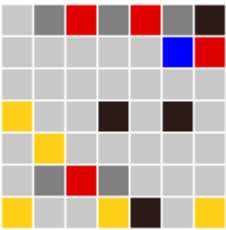
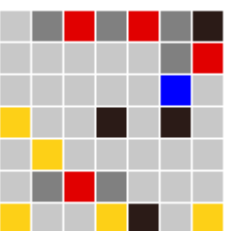
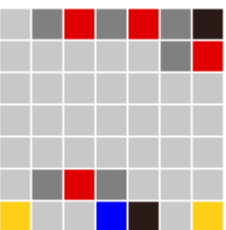
<p>This is 3th step</p> <p>Current location of agent is [1, 1]  Things at this location  [&lt;ModelBasedGoldDigger&gt;  Points: 22, Lives: 3  Moving forward  ModelBasedGoldDigger decided to move  rightwards at location: [1, 1]</p>	
<p>This is 7th step</p> <p>Current location of agent is [5, 1]  Things at this location [&lt;Tick&gt;, &lt;ModelBasedGoldDigger&gt;  Points: 18, Lives: 3  There might be bomb around this location [5, 1]  There are bombs in location [[2, 0], [6, 1]]</p>	
<p>This is 8th step</p> <p>Current location of agent is [5, 2]  Things at this location  [&lt;ModelBasedGoldDigger&gt;  Points: 22, Lives: 3  Moving forward  ModelBasedGoldDigger decided to move  rightwards at location: [5, 2]</p>	
<p>This is 34th step</p> <p>Current location of agent is [3, 6]  Things at this location [&lt;Gold&gt;, &lt;ModelBasedGoldDigger&gt;  Points: 41, Lives: 3  My points are 53 and lives are 3  ModelBasedGoldDigger collected Gold at  location: [3, 6]  I scored 50 points, I win, I won't work anymore</p>	

Table 5 - Handpicked steps and actions from model based reflex gold digger agent

Pros:

- The agent located the bombs and changes direction saving its lives and points
- Faster to reach its goal (point  $\geq 50$ )

Cons:

- If there is gold after the ticking sounds and bomb, the agent will never collect it
- If there is a bomb right below the ticking sound cell, agent will collect bomb and lose a life (example is the 3<sup>rd</sup> step in table 5, if there is a bomb at [1,1] agent will collect it)

## Goal Based Reflex Agent Type

This agent has a goal, it uses its current and past precepts to find most efficient distance to reach its goals

In our gold mine environment, the goal-based agent uses a search algorithm to find all locations of the golds present in the field, once it has got the locations, it then moves in the shortest distance possible to reach all its goals, fig 4 shows the environment and the while line path that the agent has decided to travel

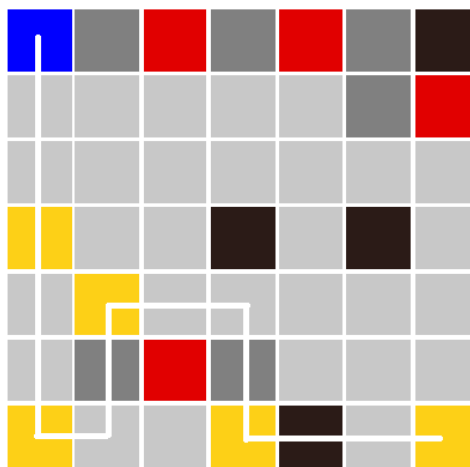


Fig 4 - Goal based gold digger agent (white line - path traversed)

The table below shows few snapshots of the goal-based agent steps, we can notice that the agent wins the game with all 3 lives and >50 points in 18 steps

<p>This is 2nd step  Current location of agent is [0, 0]  Things at this location  [&lt;GoalBasedGoldDigger&gt;]  Points: 18, Lives: 3</p>	
<p>This is 5th step  Current location of agent is [0, 3]  Things at this location [&lt;Gold&gt;, &lt;GoalBasedGoldDigger&gt;]  Points: 15, Lives: 3  My points are 27 and lives are 3  GoalBasedGoldDigger collected Gold at location: [0, 3]</p>	

<p>This is 9th step</p> <p>Current location of agent is [0, 6]  Things at this location [&lt;Gold&gt;,  &lt;GoalBasedGoldDigger&gt;]  Points: 23, Lives: 3  My points are 35 and lives are 3  GoalBasedGoldDigger collected Gold at  location: [0, 6]</p>	
<p>This is 13th step</p> <p>Current location of agent is [1, 4]  Things at this location [&lt;Gold&gt;,  &lt;GoalBasedGoldDigger&gt;]  Points: 31, Lives: 3  My points are 43 and lives are 3  GoalBasedGoldDigger collected Gold at  location: [1, 4]</p>	
<p>This is 18th step</p> <p>Current location of agent is [3, 6]  Things at this location [&lt;Gold&gt;,  &lt;GoalBasedGoldDigger&gt;]  Points: 38, Lives: 3  My points are 50 and lives are 3  GoalBasedGoldDigger collected Gold at  location: [3, 6]</p>	

Table 6 - Handpicked steps and actions from goal based reflex gold digger agent

Pros:

- Finds fastest shortest route to collect gold and win

Cons:

- Collect everything on the shortest path (gold, pebbles or bombs)
- More memory and complex computations

## Comparing the agent performances

The Table below shows how different every agent, with goal-based agent taking only 18 steps to complete the game, Simple reflex agent being dead in 7 steps and Model based agent taking the most steps (34) to win

Agent	Steps	Lives	Points	Outcome
Simple Reflex Agent	14	0	<=0	Loss/Death
Model Based Agent	34	3	>=50	Win
Goal Based Agent	18	3	>=50	Win

## Searching

Depending on the observability of our environment, the search algorithms are classified into two types:

### 1. Uninformed Search Algorithms

It is also called blind search, which means that it won't have any information other than the problem definition

Example:

- Breadth First Search
- Depth First Search
- Depth Limited Search
- Iterative Deepening Search

### 2. Informed Search Algorithms

These are also called as heuristic search strategies because they know if one goal state is more promising than the other goal state

- Breadth First Tree Search
- Depth First Tree Search
- Breadth First Search
- Depth First Graph Search
- Depth Limited Search
- Iterative Deepening Search

## Tree and Graph

The tree approach might get caught in infinite loops even in finite space, it does not keep a record of visited nodes, whereas the graph keeps a record and does not get into infinite loops

## Gold digger agent

The use of search techniques is to find the goal state given the initial state, different techniques use different algorithms, the path cost is the total sum of each hop cost, which is 1 in our case, the possible locations for initial and goal state should be bound within the X,Y of our 2D grid

The gold digger agent can move in 8 directions as shown in fig 5, the white lines show directions up, down, right, left, diagonal paths for the agent movement



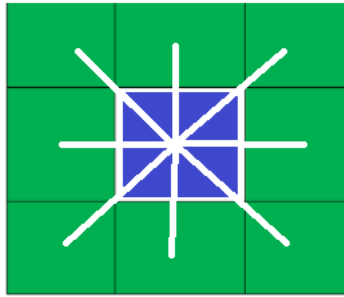


Fig 5 – Agent (colour blue) and the cells it can travel next (colour green)

The fig 6 is a 4X4 2D graph after adding edges and nodes to the graph

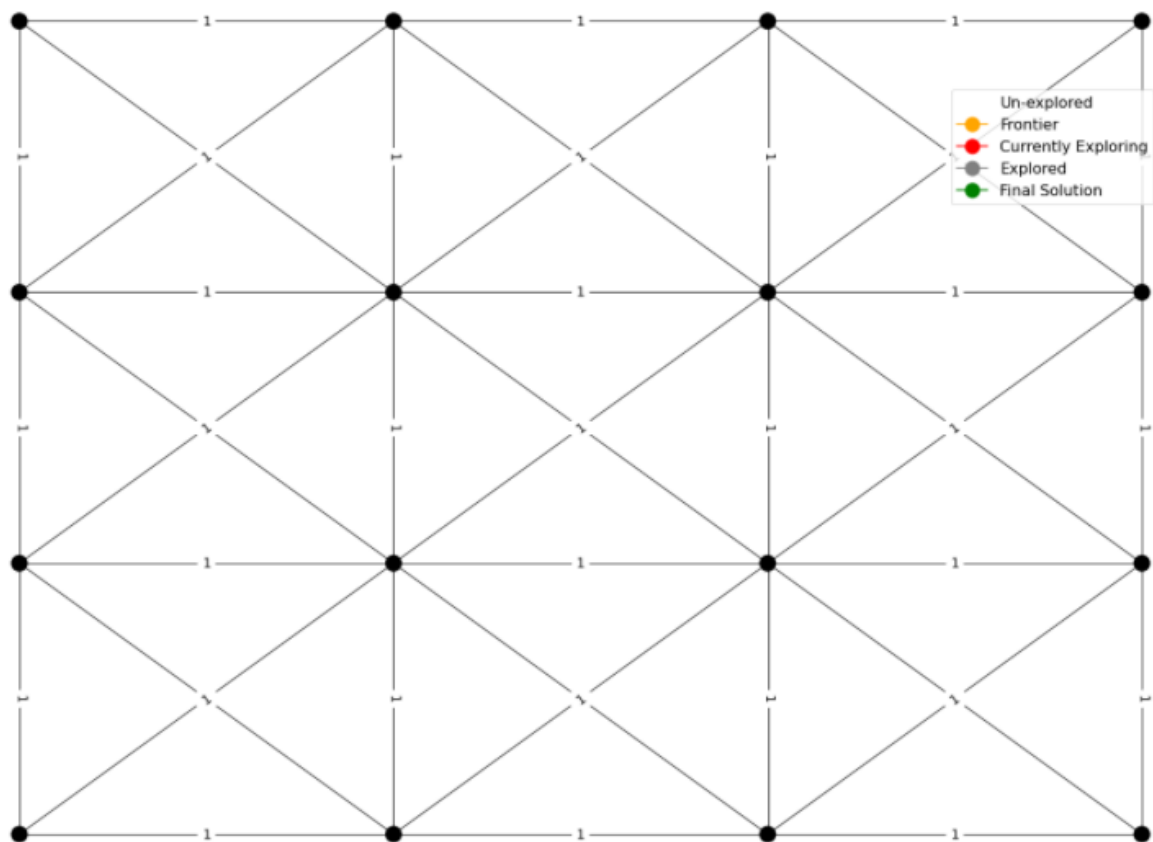


Fig 6 - Graph with nodes and edges for our gold mine environment

I have implemented six search techniques as listed below

- Breadth First Search
- Depth First Search
- Iterative Deepening Depth First Search
- Greedy Best First Search
- A star (A\*)
- Recursive Best First Search

The Breadth first search as its name depicts conducts search breadthwise, it expands all nodes at any given depth, hence it won't explore deeper nodes without exploring all nodes at current depth, BFS uses FIFO queue

Depth first search is completely opposite to Breadth first search, it explores each node until the shallowest in the current branch is reached, it uses LIFO

Iterative Deepening Search combines BFS and DFS, it goes to a certain depth and explores all nodes at that depth

Greedy best first search expands the closest node to the goal, this will work better when the step costs vary, but in our search environment, since step cost is 1 it doesn't matter (its efficient but not optimal)

A star ( $A^*$ ) compared sum of step cost and straight-line distance before deciding the next node to explore

Recursive Best First Search uses less memory, hence when memory requirements are not feasible for  $A^*$  search, RBFS becomes a better choice

The step cost in our map is 1 therefore uninformed search algorithms will perform well on our map, also since our space and time complexity is not an issue

$b$  = The branching factor, the agent can move to 8 locations from current location, so  $b = 8$

$d$  = which hop is the shallowest goal located at, depends on goal location

$m$  = The maximum depth of the search space, this depends on the 2D grid size

The Table 7 compares between six types of search techniques

Initial state is always [0,0] for our agents, here I have considered a 2D 7x7 gold mine environment

Grid Size	Goal State	Search types	Execution time	Path Cost	Nodes Explored
4X4	3_2	BFS	0.0019991397857666016s	4	80
		DFS	0	4	5
		IDDS	0	3	98
		GBFS	0.001001119613647461s	3	4
		A*	0	3	6
		RBFS	0	3	4
7x7	6_5	BFS	0.4810333251953125s	7	21821
		DFS	0	7	8
		IDDS	0.0690009593963623s	6	25639
		GBFS	0.0009996891021728516s	7	8
		A*	0	6	11
		RBFS	0	7	8
10x10	7_1	BFS	2.230025053024292s	8	128898
		DFS	0.0009770393371582031s	16	29
		IDDS	0.4329979419708252s	7	150865
		GBFS	0	7	8
		A*	0.002000093460083008s	7	22
		RBFS	0	7	8

Table 7 – search algorithm performance

Looking at the above results, I would choose RBFS for my gold mining environment to search for the gold with optimal time and space consumption