CM3038 - Artificial Intelligence for Problem Solving

Coursework #2

Darie-Dragos Mitoiu

4/24/2020

This document contains the part 2 coursework for the module called CM3038 Artificial Intelligence for Problem Solving.

Table of Contents

School of Computing Science and Digital Media
Coursework Assignment

Surname	Mitoiu
First name	Darie-Dragos
Matriculation Number	1905367
Contact phone number	07587976009
Course + Year	Computing Application Software Development Year 3
Module Co-ordinator	Dr. K. Hui
Module Number + Name	CM3038 Artificial Intelligence for Problem Solving
Coursework Title	Coursework #2
Coursework Part	Part 1 of 1
Due Date	27 April 2020 (16:00)
Feedback Due Date	25 May 2020

The University operates a Fit to Sit Policy which means that if you undertake an assessment then you are declaring yourself well enough to do so. Further details are available at: www.rgu.ac.uk/academicregulationsstudentforms

Declaration ** *This **must** be affirmed by adding your name below with the date of submission*

I acknowledge that by submitting the work, accompanied by this front cover, I take responsibility for the ownership of the submitted work.
I confirm:

- **that the work undertaken for this assignment is entirely my own and that I have not made use of any unauthorised assistance**
- **that the sources of all reference material has been properly acknowledged.**

Student Signature	Mitoiu
Date Submitted	24.04.2020

Marker's Comments	
Marker	**Grade**

** An extract from the University Regulations

6**.** **Academic Misconduct**
Refer also to Schedule 3.3 of this Regulation for guidance on this procedure.

6.1 **Academic Misconduct** is defined as any attempt by students to gain an unfair advantage in assessments and examinations. Examples of academic misconduct include plagiarism, cheating, falsifying data, collusion, bribery or attempted bribery, personation or any other activity intended to provide an unfair advantage.

(i) **Plagiarism** is the practice of presenting the thoughts or writings of another or others as original, without acknowledgement of their source(s). All material used to support a piece of work should be carefully referenced and should not normally be copied directly unless as an acknowledged quote. Text translated into the words of the individual student should in all cases acknowledge the source.

(ii) **Cheating** includes:

- the taking of any unauthorised material into an examination;
- obtaining copy of "unseen" papers in advance of an examination;
- communicating or attempting to communicate in any way with another student during an examination;
- copying or attempting to copy from another student during an examination or in the production of coursework;
- wilful deception in any element of an examination or assessment.

(iii) **Falsification of data** consists of the misrepresentation of the results of experimental work or the presentation of results from fictitious work.

(iv) **Collusion** is the representation of unauthorised group work as that of an individual student.

(v) **Bribery** is the paying, offering or attempted exchange of an inducement for information or material intended to advantage the recipient in an examination or assessment.

(vi) **Personation** consists of a substitute taking the place of a student in an examination.

A student who aids and abets a fellow student to commit academic misconduct shall be deemed to have committed academic misconduct and will be dealt with accordingly.

1. (a)

Breadth-First Search Algorithm
Nodes Explored: 289320
Solution Cost: 15.0

The Breadth-First Search Algorithm it is a searching algorithm that operates on data structures like binary trees and graphs. This algorithm allows the searching from an arbitrary node in the tree or graph and it will start exploring the nodes in the tree or graph by depth level, it will start with the current depth level and once that depth level is finished, it will move on to the next depth level as the name of the search algorithm implies, "breadth-first search algorithm".

The Breadth-First Search Algorithm it is expected to explore a large number of nodes because the searching strategy of the algorithm as it is referred by the algorithm's name, the algorithm will expand the nodes as it's searching by "breadth" in order to reach the goal. The solution cost of the path it is the same as the one found by the A* searching algorithm, assuming the A* algorithm has an admissible heuristic, the optimal solution to the problem in cause is 15, the breadth-first search algorithm having the same solution cost it means that the breadth-first was able to find the optimal solution, the breadth-first search algorithm does not ensure the finding of the optimal solution if the actions performed in order to reach the goal have different costs, the solution cost mentioned for the breadth-first search algorithm it is the result of having the goal at a favourable depth level and the breadth-first search algorithm was able to find it.

The Breadth-First Search Algorithm can guarantee that a solution it will be found based on the searching strategy if there is a solution to the problem in cause, but this algorithm cannot guarantee that the solution found it is the optimal solution to the problem in cause if the actions have different costs, if the cost of the actions it is the same for all actions, then the algorithm can guarantee the optimality of the solution.

Depth-First Search Algorithm
Nodes Explored: 1950
Solution Cost: 38.0

The Depth-First Search Algorithm it is a searching algorithm that operates on data structures like binary trees and graphs. This algorithm allows the searching from an arbitrary node in the tree or graph and it will start exploring the nodes in the tree or graph by branch depth level, it will start with the current branch depth level and once that depth level is finished, it will move on to the next depth level of the same branch and perform the same action until the end of the branch it is reached before backtracking, as the name of the search algorithm implies, “depth-first search algorithm”.

The Depth-First Search Algorithm it is expected to explore a small number of nodes because the searching strategy of the algorithm as it is referred by the algorithm’s name, the algorithm will expand the nodes as it’s searching by branch depth level in order to reach the goal. The solution cost of the path it is different of the one found by the A* searching algorithm, assuming the A* algorithm has an admissible heuristic, the optimal solution to the problem in cause is 15, the depth-first search algorithm having a different solution cost it means that the depth-first was not able to find the optimal solution, the depth-first search algorithm does not ensure the finding of the optimal solution, so this kind of result it is expected because of the searching strategy of the depth-first search algorithm.

The Depth-First Search Algorithm can guarantee that a solution it will be found based on the searching strategy if there is a solution to the problem in cause, but this algorithm cannot guarantee that the solution found it is the optimal solution to the problem in cause if the path cost to reach the goal will increase by depth level, if the goal it is situated at a lower depth level then the path cost to reach the goal it will be decreased, but this does not mean that the depth-first search algorithm it is optimal, this would be just a favourable situation in order to reach the goal in a low path cost if the goal it is situated at a low depth level.

Greedy Best First Search Algorithm
Nodes Explored: 982
Solution Cost: 15.0

The Greedy Best First Search Algorithm it is a searching algorithm that operates on data structures like binary trees and graphs. This algorithm allows the searching from an arbitrary node in the tree or graph and it will start exploring the nodes in the tree or graph by a specific rule that will guide the algorithm towards the most promising node in the tree or graph, the rule that will guide the greedy best first search algorithm it is a heuristic function that will compute a value from a specific node, this value will represent the estimated cost required to reach the goal node from the current node.

The Greedy Best First Search Algorithm it is expected to explore a small number of nodes because the searching strategy of the algorithm as it is referred by the algorithm's name, the algorithm will expand the nodes as it's searching by a "greedy" strategy which implies the use of an evaluation function using a heuristic value which is computed from the node n in the tree or graph in order to estimate the remaining cost to reach the goal. The solution cost of the path it is same as the one found by the A* searching algorithm, assuming the A* algorithm has an admissible heuristic, the optimal solution to the problem in cause is 15, the greedy best first search algorithm having the same solution cost it means that the greedy best first search algorithm was able to find the optimal solution, the greedy best first search algorithm does not ensure the finding of the optimal solution, so this kind of result would occur only when the value computed from the heuristic function it is an exact estimation in order to reach the goal.

The Greedy Best First Search Algorithm can guarantee that a solution it will be found based on the searching strategy if there is a solution to the problem in cause, but this algorithm cannot guarantee that the solution found it is the optimal solution to the problem in cause if the evaluation function which uses the heuristic function will not return an exact estimation to reach the goal from a node n in the tree or graph.

A* Search Algorithm
Nodes Explored: 3598
Solution Cost: 15.0

The A* Search Algorithm it is a searching algorithm that operates on data structures like binary trees and graphs. This algorithm allows the searching from an arbitrary node in the tree or graph and it will start exploring the nodes in the tree or graph by a specific rule that will guide the algorithm towards the most promising node in the tree or graph, the rule that will guide the A* search algorithm it is an evaluation function that will compute a value from a specific node, this value it is represented by the addition of the cost to expand the node n in cause with the estimated cost required to reach the goal node from the node n.

The A* Search Algorithm does not have an expected number of explored nodes (high or low) because of the searching strategy the algorithm it's using. The algorithm will expand the nodes as it's searching using a "guided" strategy which implies the use of an evaluation function which will return the addition of the cost of the node n that will be expanded with a heuristic value which is computed from the node n in the tree or graph in order to estimate the remaining cost to reach the goal. The solution cost of the path it is always the optimal solution for the A* Search Algorithm, assuming the A* algorithm has an admissible heuristic, the optimal solution to the problem in cause is 15 as the result found by A* algorithm implies, the A* search algorithm ensures the finding of the optimal solution if the heuristic used it is and admissible heuristic, having an admissible heuristic it means that the value computed from a node n using the heuristic function does not overestimate the actual cost to reach the goal from the node n.

The A* Search Algorithm can guarantee that a solution it will be found based on the searching strategy if there is a solution to the problem in cause, this algorithm can also ensure that an optimal solution it will be found all the time, assuming the algorithm it's using an admissible heuristic.

1. **(b)**

Student 1:

Breadth-First Search Algorithm
Nodes Explored: 289320
Solution Cost: 15.0

Student 2:

Breadth-First Search Algorithm
Nodes Explored: 287509
Solution Cost: 15.0

The Breadth-First Search Algorithm it is expected to explore a large number of nodes because the searching strategy of the algorithm as it is referred by the algorithm's name, the algorithm will expand the nodes as it's searching by "breadth" in order to reach the goal. The solution cost of the path it is the same as the one found by the A* searching algorithm for both A* search algorithm of student 1 and student 2, assuming the A* algorithm has an admissible heuristic, the optimal solution to the problem in cause is 15, the breadth-first search algorithm having the same solution cost it means that the breadth-first was able to find the optimal solution for both student 1 and student 2, the breadth-first search algorithm does not ensure the finding of the optimal solution if the actions performed in order to reach the goal have different costs, the solution cost mentioned for the breadth-first search algorithm for both student 1 and student 2 it is the result of having the goal at a favourable depth level and the breadth-first search algorithm was able to find it.

In conclusion, comparing the number of nodes explored and solution cost resulted from using the algorithm by the student 1 and student 2, there is not a relevant difference in the results of both of student 1 and student 2 to imply a poor implementation of the breadth-first search algorithm and as a result of this there is no discrepancy in the implementation of both student 1 and student 2.

Student 1:

Depth-First Search Algorithm
Nodes Explored: 1950
Solution Cost: 38.0

Student 2:

Depth-First Search Algorithm
Nodes Explored: 2500
Solution Cost: 14.0

The Depth-First Search Algorithm it is expected to explore a small number of nodes because the searching strategy of the algorithm as it is referred by the algorithm's name (a smaller number of nodes explored comparing to breadth-first search algorithm), the algorithm will expand the nodes as it's searching by branch depth level in order to reach the goal. The solution cost of the path it is different of the one found by the A* searching algorithm for both student 1 and student 2, assuming the A* algorithm has an admissible heuristic for both student 1 and student 2, the optimal solution to the problem in cause is 15, the depth-first search algorithm having a different solution cost for both student 1 and student 2, it means that the depth-first was not able to find the optimal solution, the depth-first search algorithm does not ensure the finding of the optimal solution, so this kind of result it is expected for both student 1 and student 2 because of the searching strategy of the depth-first search algorithm.

In conclusion, comparing the number of nodes explored of the depth-first search algorithm, there is not a relevant sign of poor implementation of the algorithm for both student 1 and student 2, but the solution cost of depth-first search algorithm of student 2 shows there is a sign of poor implementation for student 2 as the solution cost mentioned it is 14.0 for student 2 and the solution cost cannot be lower than 15.0 as the breadth-first and A* search algorithms imply for both student 1 and student 2, assuming the A* algorithm has an admissible heuristic.

Student 1:

Greedy Best First Search Algorithm
Nodes Explored: 982
Solution Cost: 15.0

Student 2:

Greedy Best First Search Algorithm
Nodes Explored: 7980
Solution Cost: 12.0

The Greedy Best First Search Algorithm it is expected to explore a small number of nodes because the searching strategy of the algorithm as it is referred by the algorithm's name, the algorithm will expand the nodes as it's searching by a "greedy" strategy which implies the use of an evaluation function that returns a value which is computed from the node n in the tree or graph by using a heuristic function that will allow the estimation of the remaining cost to reach the goal. The solution cost of the path it is same as the one found by the A* searching algorithm for the student 1 but a smaller solution cost is it present for student 2 which implies a poor implementation of the algorithm for student 2, assuming the A* algorithm has an admissible heuristic, the optimal solution to the problem in cause is 15, the greedy best first search algorithm having the same solution cost it means that the greedy best first search algorithm was able to find the optimal solution for student 1 but not for student 2, the greedy best first search algorithm does not ensure the finding of the optimal solution, so this kind of result would occur only when the value computed from the heuristic function it is an exact estimation in order to reach the goal only for student 1.

In conclusion, comparing the number of nodes explored and solution cost of the algorithm, there is a relevant sign of poor implementation of the algorithm for student 2 as the solution cost mentioned it is 12.0 for student 2 and the solution cost cannot be lower than 15.0 as A* search algorithms implies, assuming the A* algorithm has an admissible heuristic.

Student 1:

A* Search Algorithm
Nodes Explored: 3598
Solution Cost: 15.0

Student 2:

A* Search Algorithm
Nodes Explored: 3473
Solution Cost: 15.0

The A* Search Algorithm does not have an expected number of explored nodes (high or low) because of the searching strategy the algorithm it's using. The algorithm will expand the nodes as it's searching using a "guided" strategy which implies the use of an evaluation function which will return the result from the addition of the cost of the node n that will be expanded with a heuristic value which is computed from the node n in the tree or graph in order to estimate the remaining cost to reach the goal. The solution cost of the path it is always the optimal solution for the A* Search Algorithm, assuming the A* algorithm has an admissible heuristic, the optimal solution to the problem in cause is 15 for both student 1 and student 2, the A* search algorithm ensures the finding of the optimal solution if the heuristic used it is and admissible heuristic, having an admissible heuristic it means that the value computed from a node n using the heuristic function does not overestimate the actual cost to reach the goal from the node n.

In conclusion, comparing the number of nodes explored and solution cost of the algorithm, there is not a relevant sign of poor implementation of the algorithm for student 2 as the solution cost mentioned it is 15.0 for student 2 and the solution cost cannot be lower or higher than 15.0 as A* search algorithms implies, assuming the A* algorithm of student 1 has an admissible heuristic. As it was mentioned, there is no sign of poor implementation analysing the information given for student 2, but the heuristic could be improved to reduce the number of nodes explored for the A* algorithm.

1. **(c)**

A* Evaluation function: f(n) = g(n) + h(n)

Where:

- n = node in the fringe,
- g = cost to reach the node,
- h = heuristic function.

Ideas on eliminating the conflict when 2 or more nodes have the same f(n) value:

- Estimating the cost to reach the goal from node n in the fringe using multiple heuristic functions,
- Adjusting g(n), the cost to reach the node n in the tree,
- Adjusting h(n), the value computed from the heuristic function,
- Creating a hierarchy from the equal nodes based on the value computed from their heuristic function by passing the values computed of the nodes to a comparison function which will determine a hierarchy of the nodes based on their heuristic value.

Estimating the cost to reach the goal from node n in the fringe using multiple heuristic functions:

This technique will imply the use of different heuristic functions on the nodes in the fringe that present the same evaluation value returned from the f(n) function, each node will be passed on a separate heuristic function that will have a unique strategy of computing the estimation required to reach the goal from a node n, after the values have been computed, the node with the most accurate estimation to reach the goal from using these different heuristics will be chosen and expanded for further searching.

This technique will not present a significant drop in the number of explored nodes if the heuristic functions used in the computations of the values do not have a similar level of accuracy, if there is a significant difference in the accuracy provided by the heuristic functions, this technique might result in a very slight difference of number of nodes explored in order to reach the goal.

Adjusting g(n), the cost to reach the node n in the tree:

This technique will imply the sum of a round random number and the cost to reach the node n in the tree, this method will allow the breaking of the tie between the nodes in the fringe. The random number that will be added to the node cost has to be determined from the hash of the state in cause, for example if the state in cause represents a 2D grid, the hash value should be computed from the coordinates of the grid x and y.

This technique should solve the conflict of 2 or more nodes in the fringe presenting the same value returned from the evaluation function f(n) and reduce the number of nodes explored as the A* algorithm will no longer explore the node(s) that previously were having the same value returned from the evaluation function, the algorithm would explore only the most favourable node.

Adjusting h(n), the value computed from the heuristic function and the estimation cost to reach the goal from node n in the tree:

This technique will imply the sum of a round random number and the value returned from the heuristic function which is the estimation cost to reach the goal from a node n in the tree, this method will allow the breaking of the tie between the nodes in the fringe. The random number that will be added to the heuristic value has to be determined from the hash of the state in cause, for example if the state in cause represents a 2D grid, the hash value should be computed from the coordinates of the grid x and y.

This technique should solve the conflict of 2 or more nodes in the fringe presenting the same value returned from the evaluation function f(n) and reduce the number of nodes explored as the A* algorithm will no longer explore the node(s) that previously were having the same value returned from the evaluation function, the algorithm would explore only the most favourable node.

Hierarchy of the equal nodes based on their heuristic value:

This technique will imply the use of a comparison function which will analyse the nodes in the fringe that are presenting the same value returned from the evaluation function f(n). The analyse of the nodes will be done based on their heuristic function, the nodes will be given a descending position in a hierarchy based on the value computed from the heuristic function that the nodes are being evaluated, the node with the most accurate estimation will be chosen as the most favourable node to be further expanded.

This technique should solve the conflict of 2 or more nodes in the fringe presenting the same value returned from the evaluation function f(n) and reduce the number of nodes explored as the A* algorithm will no longer explore the node(s) that previously were having the same value returned from the evaluation function, the algorithm would explore only the most favourable node based on the selection made by the comparison function.

1. **(d) (i)**

Heuristic 1:

The mentioned heuristic will return 0 when the goal has been reached and 1 if the goal is not reached yet, this being said the heuristic it is admissible as the cost to move one person from one bank to the other bank it is 1 and this heuristic does not overestimate the actual cost to reach the goal from a node n. The performance of this heuristic would be a poor one and it will result in a larger number of explored nodes than using a heuristic which will estimate the actual cost to reach the goal from a node n.

Heuristic 2:

The mentioned heuristic will return the mismatched characters between the 2 strings, this being said the heuristic will have a better performance as it will estimate the actual cost to reach the goal from a given node n, this heuristic will always guide the A* algorithm towards the right path were the goal will be found in a small number of nodes explored, comparing to the previous heuristic, tis heuristic has a more sustainable behaviour and it fits the design of A* algorithm.

1. **(d) (ii)**

Missionaries and Cannibals State Representation: 3300S, where there are 3 missionaries and 3 cannibals in the souths, but in this representation the state presents the numbers as integers and not as a string:

Missionaries and Cannibals Heuristic:

In order to compute a value from a node n, the heuristic should identify the position of the raft and then perform the sum of the missionaries and cannibals on the bank in cause and then divide the result to the capacity of the raft, which in common "Missionaries and Cannibals" problem, the raft has a maximum capacity of 2 persons.

Example:

State to estimate from: 3300S

$h(n) = 6 / 2 = 4$

As it can be seen, the value computed from the state above it is 4, according to this strategy and the heuristic did not overestimate the cost to reach the goal from the given state, the real to reach the goal should be a 6, according to the mismatched number of persons on each bank, this heuristic has returned an optimistic value of 4.

Nota Bene:

- There will be some problems with the heuristic above, when the number that will be divided will be smaller than the capacity of the raft, the value computed will be a float one, to handle this kind of problem the result of the heuristic should be checked in order to see if it is a round number or not, if the result is not a round number it should be dealt accordingly, meaning it will be rounded before returning the value.

2. (a) (i)

In order to know if pruning can occur at node C, the alpha-beta algorithm has to be applied on the tree, the alpha-beta algorithm will start the tree exploration by looking at the node B and finding the minimum from the leaf nodes of the node B, the minimum of node B will become the number -25, the value also being updated for node B. After the algorithm has finished the searching in the node B, it will move to the node C, where it will look at the node H and it will find the value of 0, here the algorithm will check the value at node H to see if it is smaller or equal than the value of A, which now has become -25 as the searching in the node B is completed, 0 is not smaller than -25, in this case the algorithm will verify the next leaf node, which is node I, which has a value of -30, now the algorithm will perform the same verification, if -30 it is smaller or equal than -25, as this condition it is true, the algorithm will no longer look at the J node, which is the last node and it will prune it.

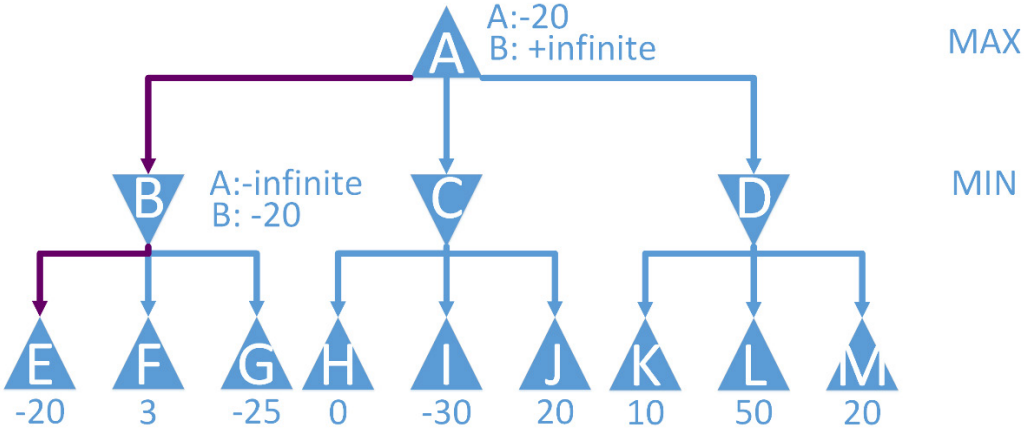
2. (a) (ii)

In order to know if pruning can occur at node D, the alpha-beta algorithm has to be applied on the tree, the alpha-beta algorithm will start the tree exploration by looking at the node B and node C and will perform the steps mentioned previously. After the same steps as mentioned above have been completed, the node A should present a value of -25, as -25 it is bigger than -30, so it is a more favourable value for the max layer, this being said, the algorithm will start searching the node D and will first look at the node K, which has a value of 10, now the algorithm will check if 10 it is smaller or equal than -25, this condition is not true and the algorithm will look at the next node without performing any pruning, the next node it is the node L, which has a value of 50, now the algorithm will perform the same verification, is 50 smaller or equal than -25, this condition is not true and the algorithm will look at the next node without performing any pruning, the last not it is the node M with a value of 20, the algorithm will check if 20 is smaller than -25, this condition is not true and the algorithm will not perform any pruning at node D which will have a value of 10 as the minimum is chosen.

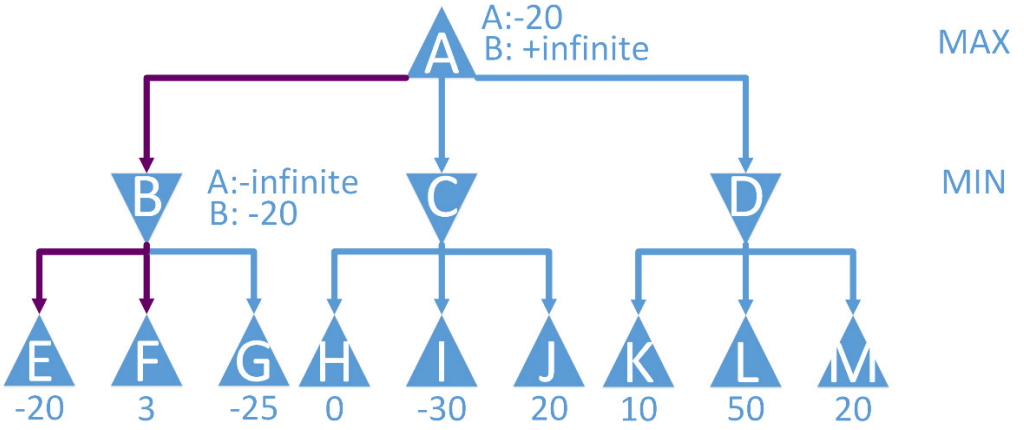
Alpha-Beta Pruning Steps Performed:

Where A = Alpha and B = Beta

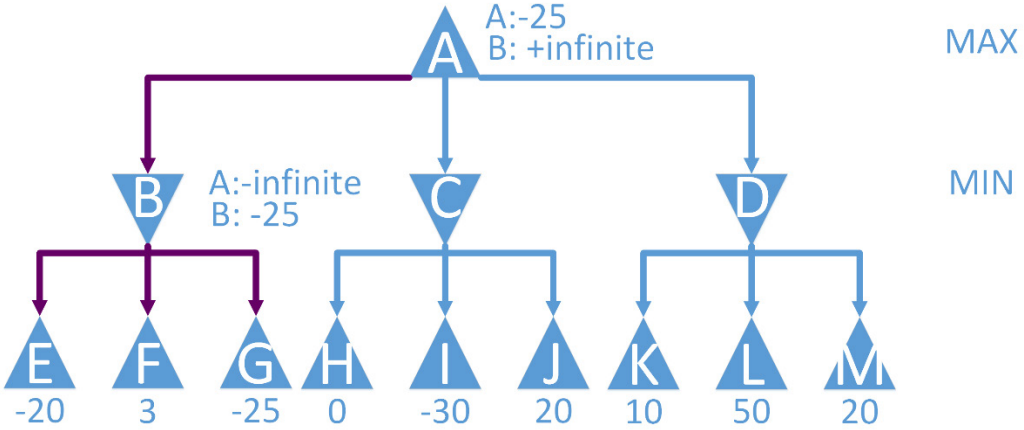
Step 1:



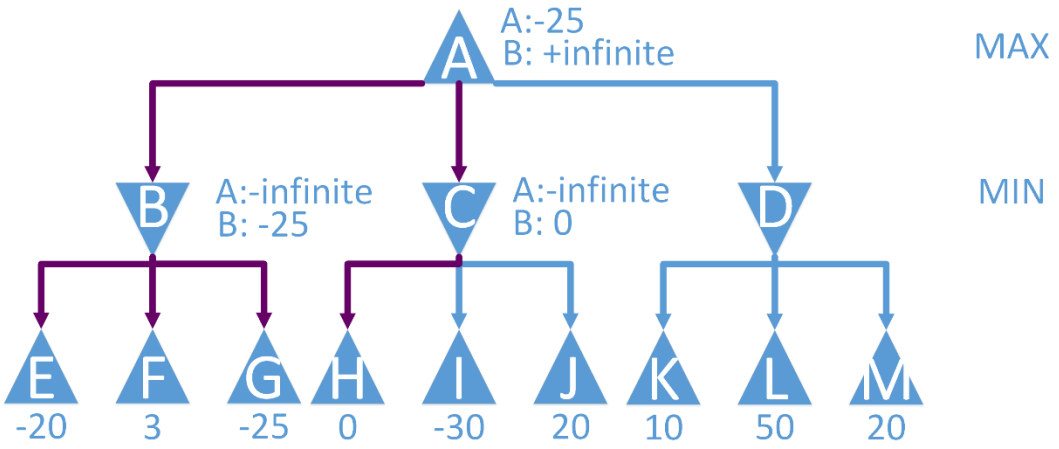
Step 2:



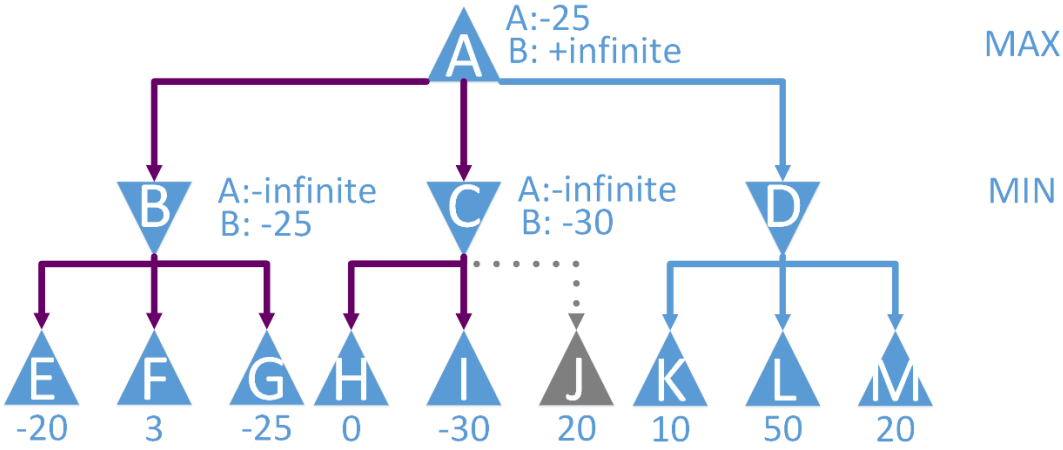
Step 3:



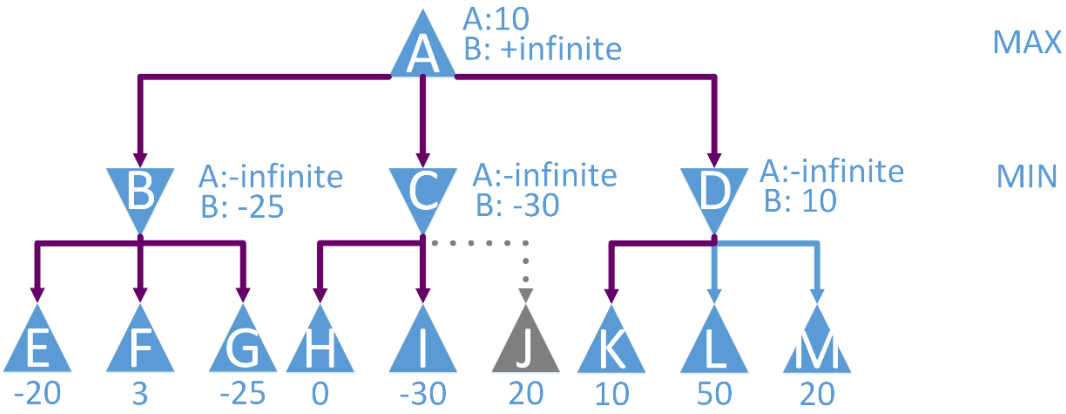
Step 4:



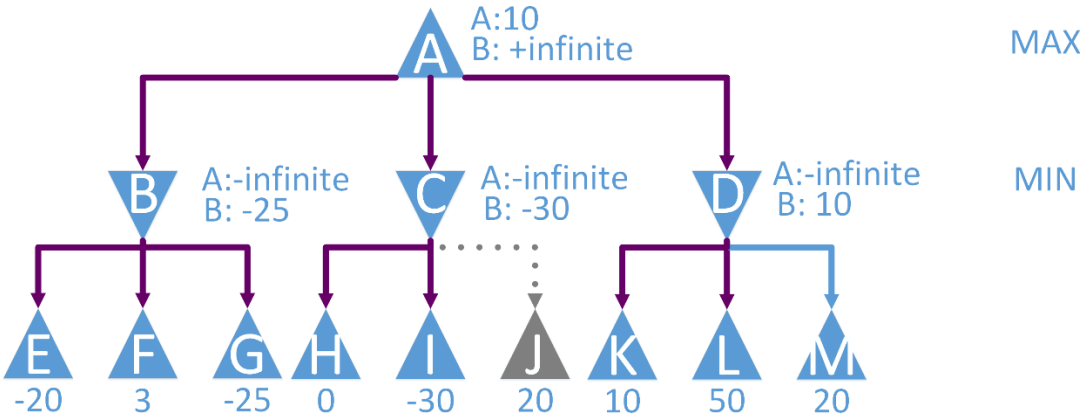
Step 5:



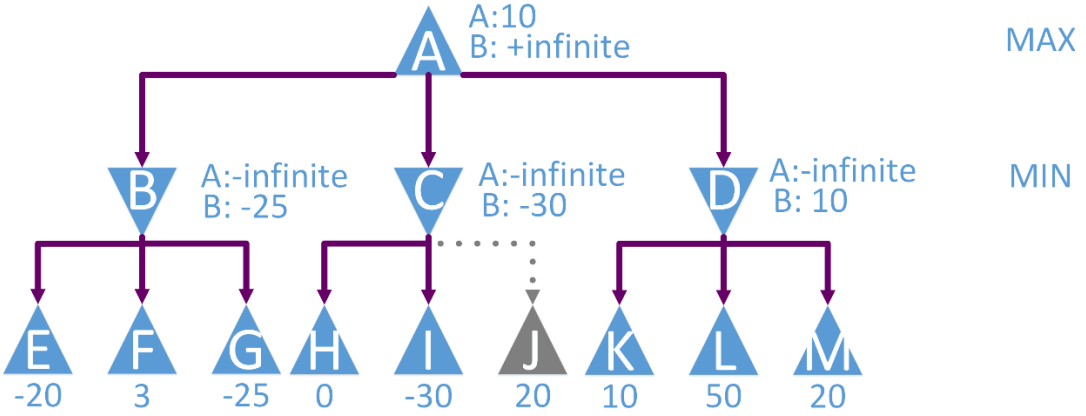
Step 6:



Step 7:


A:10
B: +infinite
MAX
B
A:-infinite
B: -25
C
A:-infinite
B: -30
D
A:-infinite
B: 10
MIN
E
F
G
H
I
J
K
L
M
-20
3
-25
0
-30
20
10
50
20

Step 8:


A:10
B: +infinite
MAX
B
A:-infinite
B: -25
C
A:-infinite
B: -30
D
A:-infinite
B: 10
MIN
E
F
G
H
I
J
K
L
M
-20
3
-25
0
-30
20
10
50
20

2. (a) (iii)

Ideas on implementing a computer player for a 2D game called "Command and Conquer":

- Adapting the Minimax algorithm for imperfect decisions in a Real-Time game using an evaluation function and a cutoff test to identify the moment when an evaluation function has to be applied,
- Adapting the Alpha-Beta algorithm for imperfect decisions in a Real-Time game using an evaluation function and a cutoff test to identify the moment when an evaluation function has to be applied,

Adapting the Minimax algorithm for imperfect information using an evaluation function and a cutoff test:

This technique will imply the use of a heuristic function which will estimate the utility value of a node which is situated at a non-terminal depth level, this process has to result in the fact that the leaves nodes will not be reached and explored and the searching process will terminate before reaching the terminal leaves and the non-terminal nodes will be treated as leaves.

This technique will imply the termination of the searching process before reaching the terminal nodes in the tree by replacing the terminal test with a cutoff test, the terminal nodes being the leaves which are situated at the terminal depth level. This technique will require the use of an evaluation function which will be applied when the cutoff test of the tree searching will occur.

The result of this technique will be one in a high use of resources in order to solve the problem in cause due to the inability of the Minimax algorithm to ignore nodes in the tree that do not offer a favourable result, a more competent algorithm that could benefit of the evaluation function and the cutoff test would be the Alpha-Beta algorithm which can solve the problem of high use of resources by ignoring nodes in the searching tree if they do not have a favourable result.

Adapting the Alpha-Beta algorithm for imperfect information using an evaluation function and a cutoff test:

This technique will imply the use of a heuristic function which will estimate the utility value of a node which is situated at a non-terminal depth level, this process has to result in the fact that the leaves nodes will not be reached and explored and the searching process will terminate before reaching the terminal leaves and the non-terminal nodes will be treated as leaves.

This technique will imply the termination of the searching process before reaching the terminal nodes in the tree by replacing the terminal test with a cutoff test, the terminal nodes being the leaves which are situated at the terminal depth level. This technique will require the use of an evaluation function which will be applied when the cutoff test of the tree searching will occur.

The implementation of this technique will result in a smaller use of resources comparing to the classic Minimax algorithm in order to solve the problem in cause due to the capacity of the Alpha-Beta algorithm to ignore nodes in the tree that do not offer a favourable result.

2. (b) (i)

Ideas of an objective function for: "Packing boxes into a cargo plane":

- Counting the number of conflicts in category of boxes,
- Sum the boxes categories and divide the number to the plane cargo capacity.

Counting the number of conflicts in category of boxes:

This idea of an objective function will analyse the state in cause and it will perform an iteration over the state containers in order to visualize the categories of the boxes, while performing the iteration a verification will be done to see if there are any boxes out of place, when a box will be found out of place, the estimation cost which will be returned by the objective function will increase for each box that is found.

Sum the boxes categories and divide the number to the plane cargo capacity:

This idea of an objective function will analyse the state in cause and it will perform an iteration over the state containers in order to visualize the categories of the boxes, while performing the iteration each category of boxes will be remembered and used latter one to find out the sum of all categories of boxes, when the sum will be identified, the result will be divided to the capacity of the plane cargo and the estimation cost will be returned by the objective function.

Ideas of an objective function for: "Scheduling classes in a university":

- Counting the number of overlapping classes.

Counting the number of overlapping classes:

This idea of an objective function will analyse the state in cause and it will perform an iteration over the state containers in order to visualize the scheduled classes, while performing the iteration each scheduled class will be remembered and used latter one to find out the duplicates of all scheduled classes, when the conflicts will be identified, the estimation cost will be returned by the objective function.

Ideas of an objective function for: "Assigning deliveries to a fleet of couriers":

- Counting the number of overlapping deliveries of couriers.

Counting the number of overlapping deliveries of couriers:

This idea of an objective function will analyse the state in cause and it will perform an iteration over the state containers in order to visualize the scheduled deliveries which will be required to be performed by the couriers, while performing the iteration each delivery of the couriers will be remembered and used latter one to find out the duplicates of all deliveries or if one delivery it is assigned to more than one courier, when the conflicts will be identified, the estimation cost will be returned by the objective function.

2. (b) (ii)

Local Search Algorithm:

The local search algorithm it is a type of an algorithm that uses a heuristic method to solve problems that require the optimal solution to be found. The local search algorithm explores the nodes by searching from one solution to another in the searching space where the candidate solutions are to be found by applying local changes until an optimal solution is found.

The local search algorithm does not remember the path it is taking in order to find the optimal solution as the algorithm it is interested only in finding the goal state or a better state from the current one by looking at the local neighbourhood in the state, this being searching space where the candidate solutions are to be found.

Genetic Algorithm:

The genetic algorithm it is a type of an algorithm inspired by the process of natural selection, the algorithm having as scope the provision of good quality solution to optimization and search problems by using concepts known in biology.

The genetic algorithm has a very important element which is called population, in a genetic algorithm the element called population it is used to refer to possible candidate solutions and each candidate solution presents a list of attributes, this are the individual chromosomes or genotype which can be modified and last but not the least the possible candidate solutions are represented as a binary string of 1's and 0's.

The genetic algorithm in order to be used for an optimization of a problem, the algorithm must present a genetic representation of the solution domain and a fitness function which will allow the evaluation of the solution domain.

Discuss the statement: "Local search is a genetic algorithm where the population size is 1."

The genetic algorithm it is referred as an optimization technique that uses the population of possible candidate solutions and then it will start exploring the search space where the candidate solution which are present by evolving the population through the steps of parent selection, crossover, mutation and replacement.

The local search algorithm it is only concerned about reaching a better state from the current one that the algorithm it is situated at or even the goal state if all the optimal selections have been made previously.

This being said, the local search algorithm will look search by exploring the candidate solutions in the searching space one solution at the time and the solution will be chosen only if it is better than the current state, the genetic algorithm on the other side it acts as optimization technique for the local search algorithm where the objective it is to evolve the population in the searching space of possible candidate solutions.

Local Search algorithm and Genetic Algorithm Similarities:

- Both algorithms use a searching space in order to find the optimal solution,
- Both algorithms are concerned in finding the optimal solution.

Local Search algorithm and Genetic Algorithm Differences:

- The genetic algorithm it is an optimization technique for searching algorithms,
- The local search does not present a fitness function,
- The local search does not present a genetic representation of the solution domain.

3. **(a)**

Relevant features to the problem in cause:

- Symptom,
- Age,
- Gender,
- Health conditions,
- Job related activity.
- Undertaken surgeries.

Relevant features to the solution in cause:

- Diagnosis,
- Exercises types,
- Number of repetitions of each exercise,
- Minimum number of steps required.

3. **(b)**

Selected features of the problem domain:

- Symptom,
- Health conditions,
- Job related activity,
- Undertaken surgeries,
- Age.

In order to calculate the similarity between the new problem and the previous cases, the following similarity techniques must be used:

- Local similarity,
- Global similarity

The similarity of the new problem and the existent case in the database will be done using the local similarity function which takes as an input the new problem and a case from the database.

The finalization of the computed value will be done using the global similarity formula which takes as argument the new problem and a case from the database and also makes use of the local similarity function in order to return the final similarity value.

3. **(c)**

Example of new problem:

Symptom: Knee Pain,
Health Conditions: None,
Job Activity: Office activity,
Surgery: Knee replacement surgery,
Age: 45,
Gender: Female

Example of case and solution in database:

Symptom: Knee Pain,
Health Conditions: None,
Job Activity: Office activity,
Surgery: Knee replacement surgery,
Age: 50,
Gender: Female

Case Solution:

Diagnosis: Tendonitis,
Exercise 1 type: Straight-leg raises to the front,
Exercise 2 type: Terminal knee extension,
Exercise 1 repetitions: 8-12 times,
Exercise 2 repetitions: 8-12 times,
Minimum walking steps: 3000

In order to calculate the similarity of the new problem and the case in the database, the local similarity formula must be used in order to identify the similarity between the new problem features and the features of the case into the database, once these similarities are identified, the global similarity formula must be used in order to identify the final similarity between the new problem and the case in the database, this global similarity formula will make use of the local values obtained using the local similarity formula.

3. **(d)**

Acquiring the medical knowledge of real-world health situations:

- Accessing an API designed for real-world health situations and storing the relevant data in the database.
- Creating a Web Crawler (Spiderbot) type of software and retrieve information from specialized websites in order to create and store the relevant medical information in the database.

3. **(e)**

Arguments for a GP which may be concerned of the result provided by the system in cause:

- Explain the calculation method, meaning the way the similarity value it is computed from the new problem and previous cases in the database.
- Explain the changes of a self-management plan to succeed based on previous cases by retrieving the relevant information from database and analysing the data using a machine learning technique like the random forest model or the logistic regression model.

4. **(a) (i)**

The main steps resulted from combining the Local Search Algorithm and a Genetic Algorithm:

- Initialize population (candidate solution in the searching space),
- Start searching from a candidate solution in the searching space,
- Iterate over the neighbours in the searching space in order to find the next optimal next candidate solution,
- Select parents,
- Crossover,
- Mutate,
- Update population (candidate solutions in searching space),
- Output best candidate solution.

4. (a) (ii)

Advantages of combined strategy:

- The search optimization provided by the genetic algorithm will result in a fast and efficient search hybrid algorithm, fast and efficient meaning the solution will be provided in less time,
- The mutation of the population (candidate solutions) will result in the extended number of candidate solutions,
- More favourable candidate solutions due to the genetic algorithm workflow.

Disadvantages of combined strategy:

- Genetic algorithm alone will always find an optimal solution and does need to be combined to solve the same problem,
- The local search algorithm it is only interested in the neighbour candidate solution, while the genetic algorithm it is interested in all the population (candidate solutions).

4. (b) (i)

Fitness Proportionate Selection of Solution S:

Probability = $\frac{15}{5+40+20+15+20} = \frac{15}{100} = 0.15\%$

4. (b) (ii)

Number of children produced under one-point crossover: 2

4. (b) (iii)

The mentioned genes will be retained in order to allow the population to be updated towards the most favourable candidate solution, the genes are retained because they are present in the candidate solutions that present the most favourable probability of selection given by the fitness function while the population is being updated.

4. (c)

Usage of the Genetic Algorithm:

- The Genetic Algorithm should be used to solve problems where the evaluation of all possible candidate solutions it is infeasible.
- The Genetic Algorithm should be used when dealing with a hill-climbing type of problem, a local search type of algorithm would be right approach.
- The Genetic Algorithm should be used only when a competent fitness function it is present.

4. (d)

Case-Based Reasoning Approach:

The case-based reasoning approach can be used to the problem in cause as a database of previous items sold associated with their selling location and other natural factors like the regional soil conditions and local climate patterns, this method will allow the analysis of a new item which will be required to be sold to be analysed against previous similar items in the database and decide the relevant details of selling the item, like the location where the item should sold and the number of items that should be retained for a particular store of the business based on previous cases of similar items sold at similar locations.

References

1. K.Hui (2020), “Artificial Intelligence for Problem Solving (CM3038) Lecture #2 Uninformed Search”, Link: http://campusmoodle.rgu.ac.uk/pluginfile.php/4980445/mod_folder/content/0/02%20Uninformed%20Search.pdf?forcedownload=1 ,
2. K.Hui (2020), “Artificial Intelligence for Problem Solving (CM3038) Lecture #3 More Uninformed Searches”, Link: http://campusmoodle.rgu.ac.uk/pluginfile.php/4980448/mod_folder/content/0/03%20More%20Uninformed%20Searches.pdf?forcedownload=1 ,
3. K.Hui (2020), “Artificial Intelligence for Problem Solving (CM3038) Lecture #4 Informed Search”, Link: http://campusmoodle.rgu.ac.uk/pluginfile.php/4980449/mod_folder/content/0/04%20Informed%20Search.pdf?forcedownload=1 ,
4. K.Hui (2020), “Artificial Intelligence for Problem Solving (CM3038) Lecture #5 More Informed Search”, Link: http://campusmoodle.rgu.ac.uk/pluginfile.php/4980451/mod_folder/content/0/05%20More%20Informed%20Search.pdf?forcedownload=1 ,
5. K.Hui (2020), “Artificial Intelligence for Problem Solving (CM3038) Lecture #6 Adversarial Search”, Link: http://campusmoodle.rgu.ac.uk/pluginfile.php/4980453/mod_folder/content/0/06%20Adversarial%20Search.pdf?forcedownload=1 ,
6. K.Hui (2020), “Artificial Intelligence for Problem Solving (CM3038) Lecture #7 Local Search”, Link: http://campusmoodle.rgu.ac.uk/pluginfile.php/4980456/mod_folder/content/0/07%20Local%20Search.pdf?forcedownload=1 ,
7. Nirmalie Wiratunga (2020) “Artificial Intelligence for Problem Solving (CM3038) Lecture 1: Introduction”, Link: http://campusmoodle.rgu.ac.uk/mod/resource/view.php?id=3488713 ,
8. Nirmalie Wiratunga (2020) “Artificial Intelligence for Problem Solving (CM3038) Lecture 2: Case Retrieval”, Link: http://campusmoodle.rgu.ac.uk/mod/resource/view.php?id=3488714 ,
9. John McCall, Ciprian Zavoianu (2020), “Artificial Intelligence for Problem Solving (CM3038) Genetic Algorithms”, Link: http://campusmoodle.rgu.ac.uk/mod/resource/view.php?id=3488710 ,
10. K.Hui (2020), “Artificial Intelligence for Problem Solving (CM3038) Laboratory #2 Solution/Discussion” (Version 1.0) [source code] Link: http://campusmoodle.rgu.ac.uk/pluginfile.php/4963699/mod_folder/content/0/Lab-02-code.zip?forcedownload=1 ,
11. K.Hui (2020), “Artificial Intelligence for Problem Solving (CM3038) Laboratory #5 Solution/Discussion” (Version 1.0) [source code] Link: http://campusmoodle.rgu.ac.uk/pluginfile.php/4963728/mod_folder/content/0/Lab05-code.zip?forcedownload=1