

MSc in Software Engineering and Database Technologies

CT621 Artificial Intelligence

**Student Name:** Cristina Borges

**NUIG ID Number:** 20234955

**Course:** CT621 Artificial Intelligence

**Workshop No:** Week 2

**Assignments:** Week 2

**Date of Submission:** 30/05/2021

Contents

1.0 Assignment 1 2

1.1 Question 1 3

1.2 Question 2 4

1.3 Question 3 5

1.4 Question 4 6

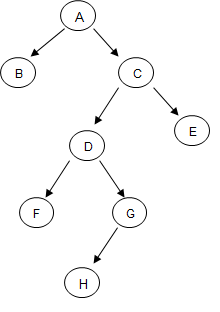
# Assignment

**Question 1:**

Write a short essay that describes heuristic-based search (Minimum 800 words). In your answer describe a scenario where you may opt to use a heuristic-based search algorithm over an exhaustive approach. Include in your answer a reason why the choice of heuristic may impact the success of the algorithm. Also give examples of the types of heuristics that you might adopt for such a problem and describe why some of these heuristics are less suitable than others.

**Question 2:**

Show all steps involved in searching the tree shown below using an iterative-deepening search. Also, would you recommend that an iterative-deepening search algorithm be selected over a depth-first search in this case? Does this answer hold for all cases?



**Question 3:**

In the 8-Queen Puzzle shown in Workshop 2 Section 2 we represent a particular state of the problem as a set of numerical digits, so that each digit specifies the row occupied by a queen in a particular column. Therefore [8, 2, 4, 6, 1, 8, 5, 3] would represent a configuration where the queen in the first column occupies row 8, the queen in the second column occupies row 2, the queen in the third column occupies row 4, and so on.

Now consider a very small population consisting of the following five individuals:

[8, 2, 4, 6, 1, 8, 5, 1]

[2, 3, 1, 6, 1, 2, 6, 6]

[1, 6, 4, 7, 2, 3, 5, 3]

[4, 1, 1, 7, 8, 8, 2, 1]

[1, 3, 2, 8, 6, 7, 5, 4]

Show how you would evolve the population using the EVOLVE\_POPULATION function outlined the in Workshop 2 Section 2 slides. Your output should be a new generation of individuals. It is important to fully show your workings. For example, when you perform the REPRODUCE function you should clearly show the crossover point you choose and how the resulting child individual is a combination of the two chosen parents.

## 1.1 Question 1

**What is Heuristic-based search**

A heuristic-based search is a technique that is used in AI as a means of solving problems faster. The process followed by a heuristic search can be described as one where it looks at the search at each branching step to identify the most promising direction to follow (Randall). The outcome of this method is that it will search for the best possible solution that can be found in a reasonable amount of time, however, it is unlikely to be the best solution that could have been achieved meaning the improvement in efficiency has come as the result of a trade off in completeness (Dataflair).

**Heuristic search vs exhaustive search**

Exhaustive search is, as the name suggests a method of searching which looks at all possible combinations to find a solution. As the exhaustive search looks at all possible nodes, the solution it finds would be the globally optimal solution to a problem rather than one that would be considered locally optimal only such as via the heuristic search (Dr M Humphrys).

Generally exhaustive searches are used for problems with a smaller solution space, so they are best suited to discrete problems. Exhaustive (or brute force) algorithms are also used in the area of network security where malicious actors may make sue of brute-force attempts to access protected systems or information (G De Luca)

The most significant problem experienced when using exhaustive searches is how poorly they scale in terms of both memory use and performance. Using an exhaustive search for the N-Queens problem shows that the time taken hits a wall when you reach the 10-queen mark (jboss.org), however, Sosic and Gu showed that solutions could be find in reasonable times for a significantly higher number of queens.

**Choice of heuristic impacting success, examples to adopt & when are they less suitable**

There are a number of different heuristic search techniques that users can choose from depending on their objectives which include:

1. Direct heuristic search
   1. A\*Search
   2. Greedy Best First Search
2. Weak heuristic search
   1. Breadth First Search
   2. Uniform Cost Search
   3. Depth First Search
   4. Bidirectional Search
3. Hill Climbing in AI (Simple, Steepest Ascent or Stochastic)

While this isn’t a comprehensive list, it does give an idea of the range of choices available. Each heuristic also has their own advantages and disadvantages which can have an impact on their success and drive when you might want to use each one.

A\* Search is considered one of the better algorithms for pathfinding but won’t get the optimal result for example (Geeksforgeeks).

Greedy best first search is a variant of the best search first technique which is easy to create and have shorter run times, however, due to the decision making structure of the greedy technique, you only get one opportunity to arrive at an optimal solution and it may often be the case that the algorithm isn't actually correct (Hackerearth).

So the direct heuristic searches aren’t the best choice if you require the globally optimal solution rather than the optimal solution based on the current node.

In the case of the weak heuristic (uninformed search), the factors that could make the heuristic less suitable and impact its success include vary depend on the search type used.

Breadth first search will rarely follow the wrong path for long, however, it does have a very hight time complexity and it has very long pathways (BrainKart.com)

Using uniform cost search will look for the path with the lowest cumulative cost, however, storage requirements are high, and the algorithm can enter infinite loops (S Jagga)

Depth first search is more efficient in terms of memory and will reach a result faster than breadth first search, however, it is possible that the final destination node is never achieved (BrainKart.com)

Bidirectional search is a form of brute force search which means it will not be efficient when you compare it to the other algorithms available.

The hill climbing search algorithm is a type of heuristic search that looks for stats that are an improvement on its current state and terminates when no better option is available. The main issues encountered with this approach are the failure to find a global maximum and instead solving for a local maximum only. It also doesn’t recall past states meaning backtracking isn’t possible (Techvidvan).

**Conclusions**

Unless you are doing something, which requires the globally optimal solution, the use of heuristic searches seem to make far more sense than an exhaustive search due to the efficiencies gained in terms of overheads, time as well as less general complexity. For a lot of functions, the loss of accuracy in the answer is also unlikely to be material.

## 1.2 Question 2

Steps to follow would be to:

1. Specify starting point
2. Specify nodes at each depth
   1. Depth 0 contains A
   2. Depth 1 contains B & C
   3. Depth 2 contains D & E
   4. Depth 3 contains F & G
   5. Depth 4 contains H
3. Specify depth bound for each iteration
4. Specify goal node (will look at solutions where this is varied
5. Quote order of expansion

Iteration 1

Max depth bound set to 0

A is at max depth bound

Order of expansion is

A

Iteration 2

Max depth bound set to 1

B and C are at max depth bound

Order of expansion is

ABC

Iteration 3

Max depth bound set to 2

D and E are at max depth bound

Order of expansion is

ABCDE

Iteration 4

Max depth bound set to 3

F and G are at max depth bound

Order of expansion is

ABCDFGE

Iteration 5

Max depth bound set to 4

H is at max depth bound

Order of expansion is

ABCDFGHE

Overall iterative deepening search gives a total path of

A ABC ABCDE ABCDFGE ABCDFGHE

Looking at the full expansion of paths for each iteration, to find the path to a goal node, you only need to find the first mention of that node

Iterative deepening search would be preferred for trees with large state spaces or with branching factor of greater than 2.

Iterative deepening search will find the optimal solution first

Due to the size of the tree, I would be indifferent between the depth first search and the iterative deepening search. Depth first search would only arrive at the goal quicker if we were looking to reach node H.

The shape of the tree also means that the iterative deepening depth first search will follow the same solution as the breadth first search.

As the number of levels increases, the depth first search would become a more attractive option.

## 1.3 Question 3

The process of evolving a population is explained by the following

*Function: EVOLVE\_POPULATION*

*Population is a set of individuals*

*HEURISTIC measure of fitness*

*new\_population <- empty\_set*

***for*** *i=1 to SIZE(population)* ***do***

*x <- RANDOM\_SELECTION (population, HEURISTIC)*

*y <- RANDOM\_SELECTION (population, HEURISTIC)*

*child <- REPRODUCE (x,y)*

***if (****small random probability)*

***then*** *child <- MUTATION (child)*

*add child to new\_population*

*population <- new\_population*

In the example above, it selects 2 individuals from the population at random

2 individuals are recombined using the reproduce function

Steps to follow for question

**Calculate Fitness**

By analysis of each individual set, I identified the number of attacking/non-attacking pairs for each:

1. **Attacking 4**

**Non-attacking 24**

1. **Attacking 8**

**Non-attacking 20**

1. **Attacking 4**

**Non-attacking 24**

1. **Attacking 8**

**Non-attacking 20**

1. **Attacking 7**

**Non-attacking 21**

The number of non-attacking pairs for each is the fitness score. Fitness score is then used to calculate the probability of **random selection for each set.**

**Random Selection**

**Sum of fitness scores = 24 + 20 + 24 + 20 + 21 = 109**

**Set 1 24 / 109 = 0.22**

**Set 2 20 / 109 = 0.18**

**Set 3 24 / 109 = 0.22**

**Set 4 20 / 109 = 0.18**

**Set 5 21 / 109 = 0.20**

**For the purpose of progressing the question, I have assumed the sets with the highest probability are selected i.e. Set 1 & Set 3.**

**Reproduce**

**The next step will involve the combination of these two sets.**

**I used random.org to decide on the position within the string for the crossover point. Crossover takes place after entry 3 – I have colour coded below**

**Set 1**

**[8, 2, 4, 6,1, 8, 5,1]**

**Set 3**

**[1, 6, 4, 7, 2, 3, 5, 3]**

**Child Set**

**[8, 2, 4, 7, 2, 3, 5, 3]**

**Appendix 1**

**Assignment 1**

**References**

Randall M (2020) Heuristic Search Part 1: Introduction and Basic Search Methods

Available at: <https://www.lancaster.ac.uk/stor-i-student-sites/matthew-randall/2020/04/01/heuristic-search-part-1-introduction-and-basic-search-methods/>

Accessed 30th May 2021

DataFlair (2020) What is Heuristic Search – Techniques & Hill Climbing in AI

Available at: <https://data-flair.training/blogs/heuristic-search-ai/>

Accessed 30th May 2021

javaTpoint (undated) Informed Search Algorithms

Available at: <https://www.javatpoint.com/ai-informed-search-algorithms>

Accessed 30th May 2021

Humphrys Dr M. (undated) Defining AI as “search”

Available at: https://computing.dcu.ie/~humphrys/Notes/AI/search.html

Accessed 30th May 2021

De Luca G (2020) Brute Force Algorithm in Cybersecurity and String Search

Available at: <https://www.baeldung.com/cs/brute-force-cybersecurity-string-search>

Accessed 30th May 2021

Jboss.org (undated) Chapter 8. Exhaustive search

Available at: <https://docs.optaplanner.org/6.2.0.CR3/optaplanner-docs/html/exhaustiveSearch.html>

Accessed 30th May 2021

Sosic J & Gu Ju (1990) A Polynomial Time Algorithm for the N-Queens Problem

Available at: https://fizyka.umk.pl/~milosz/AlgIILab/10.1.1.57.4685.pdf

Accessed 30th May 2021

[dave@cs.cf.ac.uk](mailto:dave@cs.cf.ac.uk) (undated) Heuristic Search

Available at: <https://users.cs.cf.ac.uk/Dave.Marshall/AI2/node23.html>

Accessed 30th May 2021

GeeksforGeeks (2021) A\*Search Algorithm

Available at:<https://www.geeksforgeeks.org/a-search-algorithm/>

Accessed 30th May 2021

Hackerearth (undated) Basics of Greedy Algorithms

Available at: <https://www.mygreatlearning.com/blog/best-first-search-bfs/#:~:text=Best%20first%20search%20uses%20the,state%20via%20the%20shortest%20path>.

Accessed 30th May 2021

BrainKart.com (undated) Uninformed Search

Available at: <https://www.brainkart.com/article/Uninformed-Search_8875/>

Accessed 30th May 2021

Jaggi S (undated) Uniform Cost Search

Available at: <https://www.educba.com/uniform-cost-search/>

Accessed 30th May 2021

Techvidvan (undated) Heuristic Search Techniques in Artificial Intelligence

Available at: <https://techvidvan.com/tutorials/ai-heuristic-search/>

Accessed 30th May 2021

Question 3

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Set 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 |  |  |  |  | X |  |  | X |
| 2 |  | X |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |
| 4 |  |  | X |  |  |  |  |  |
| 5 |  |  |  |  |  |  | X |  |
| 6 |  |  |  | X |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |
| 8 | X |  |  |  |  | X |  |  |

24 non-attacking pairs

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Set 2 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 |  |  | X |  | X |  |  |  |
| 2 | X |  |  |  |  | X |  |  |
| 3 |  | X |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |
| 6 |  |  |  | X |  |  | X | X |
| 7 |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |  |

20 non-attacking pairs

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Set 3 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | X |  |  |  |  |  |  | X |
| 2 |  |  |  |  |  |  | X |  |
| 3 |  |  |  |  |  |  |  |  |
| 4 |  |  | X |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |
| 6 |  | X |  |  |  |  |  |  |
| 7 |  |  |  | X |  |  |  |  |
| 8 |  |  |  |  | X | X |  |  |

24 non-attacking pairs

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Set 4 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 |  | X | X |  |  |  |  | X |
| 2 |  |  |  |  |  |  | X |  |
| 3 |  |  |  |  |  |  |  |  |
| 4 | X |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |
| 7 |  |  |  | X |  |  |  |  |
| 8 |  |  |  |  | X | X |  |  |

20 non-attacking pairs

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Set 5 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | X |  |  |  |  |  |  |  |
| 2 |  |  | X |  |  |  |  |  |
| 3 |  | X |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  | X |
| 5 |  |  |  |  |  |  | X |  |
| 6 |  |  |  |  | X |  |  |  |
| 7 |  |  |  |  |  | X |  |  |
| 8 |  |  |  | X |  |  |  |  |

21 non-attacking pairs