**Mini Project**

**Tents and Trees puzzle**

- for Introduction to Artificial Intelligence 1 WS23/24

Ein Bild, das Screenshot, Grün, Quadrat, Cartoon enthält.

Automatisch generierte Beschreibung

Figure 1: Tents and Trees puzzle finished Mini Project result[[1]](#footnote-1)

University Klagenfurt

M.Sc. Artificial Intelligence & Cybersecurity

Dimitra Brountsou

Matr.Nr.: 12128450

Contents

[1. Problem Statement – Tents and Trees puzzle 3](#_Toc160056662)

[2. Task of the project 3](#_Toc160056663)

[3. Implementation 4](#_Toc160056664)

[3.1 Considerationson choosing the method 4](#_Toc160056665)

[3.2 Requirements for compiling the code 5](#_Toc160056666)

[3.3 Implementation & testing 8](#_Toc160056667)

[4. Consideration on ethics 13](#_Toc160056668)

List of figures

[Figure 1: Tents and Trees puzzle finished Mini Project result 1](#_Toc159965662)

# 1. Problem Statement – Tents and Trees puzzle

Artificial intelligence (AI) is a hot topic nowadays and one of the areas where its advancements will define our future. Generating robust algorithms in AI, especially for complex problems, is essential. These robust algorithms are best tested on settings that are easy to interpret and understand in terms of space and time complexity.

As part of the Introduction to AI course project at Klagenfurt University, the main task was to build an automated Puzzle game with a search algorithms to apply the theoretical understanding of AI algorithms to a relatively simple game.

The goal of the puzzle game "Tents and Trees" is to place tents in a grid full of trees according to certain rules. Each puzzle has a grid that represents a campsite with sparse trees. The goals of setting up a tent are:

1. Have exactly one tent per tree (horizontally or vertically, not diagonally).

2. Tents may not be adjacent horizontally, vertically or diagonally.

The challenge is to strategically decide where to pitch your tent based on the information provided and following the rules. The game is over if the player clicks on all cells in which a tent has to be assign as per rules to a tree.

# 2. Task of the project

The main task of the project was to implement an automated Tents and Trees puzzle game using at least one of the search Algorithms covered in the Intro to AI module. Also, the agent has to run on an Ubuntu 24.04.3 LTS.

We should also write an essay, which documents our agents and the development thereof. In particular, it should contain considerations on choosing the method(s), and on design, implementation, and testing. The essay should also contain the following additional consideration on ethics: Imagine that a variant of your agent will be deployed not in a computer game, but in reality. Starting from this scenario, discuss the deployment of AI methods in military applications from an ethical perspective.

# 3. Implementation

## 3.1 Considerationson choosing the method

Preparation:

Before actually starting the project, it is important, to understand what exactly is required. To understand what you have to deal with, it is good to have a look on the following points:

• Tools required

• Defining inputs, outputs, Class and function definitions

• Coming up with timeline for the project and sticking to it

• Try to find the right searching algorithm

• Testing and improvement

• Results and Conclusions

The way that was found for this project can play the Minesweeper game manually or automatically with an automated game player. Also, for this project the main focus was on the implementation of propositional Logic.

The design phase is crucial to have a stable implementation, where you can then build/extend on it. In the design phase you have, among other things, to choose a right agent as an first important step.

You can choose to use a simple agent, whose behaviors are based on a simple mapping from states to actions. Yet in this case, you must take into account the fact that such agents cannot function well in contexts where this mapping would be too large to store and would take too much time to learn. On the other hand, you can use goal-based agents that take into account potential future actions and the desirability of their results. Problem-solving (one kind of goal-based agent) agents think about the world using atomic representations.

As a result, states of the world are viewed as wholes with no internal structure that the algorithms for solving problems can see. Planning agents are goal-based agents that use more advanced factored or structured representations. Here, by properly describing the components of a "problem" and its "solution," and by providing several instances to demonstrate these definitions, it is made abundantly evident that problem solving must be taken into account during the development phase. The next stage is to evaluate the most effective algorithms after describing a number of all-purpose search algorithms that can be employed to address these issues.

The evaluation of your algorithm should also be considered in the design phase. The main metric for evaluating an algorithm will be the percentage of games won, or win ratio. The algorithms can also be tested on different levels (like implemented: level easy, medium, hard, extreme) and the expected number of guesses for each algorithm has also to be considered for the algorithm choice. The ideal solver should have a high success rate and a low guess average. Mean guesses alone, however, do not necessarily represent a solver's performance.

## 3.2 Requirements for compiling the code

The process of looking for a sequence of actions that reaches the goal is called search. A search algorithm takes a problem as input and returns a solution in the form of an action sequence. Once a solution is found, the actions it recommends can be carried out.

In the following the most important implementation challenges for an automated minesweeper solving algorithm will be mentioned.

But before we come to the challenges, the meaning of the agent must be specified. So, agents offer a powerful method for game modeling and optimization by enabling the execution of numerous potential scenarios for discovering the best viable strategy. To maximize the goals set out by the designers while also adhering to the predetermined limits, agents can automate tuning and close the testing loop on updates. Because of the present state of the art in computing, all types of AI algorithms can be used to power the suggested agents, allowing search-based methodologies to be used right from the start of the design process. AI-powered agents are anticipated to become a necessary tool in the future.

To get back to the most important implementation challenges, the focus in the development phase has to be on the first move of a minesweeper game because the first click is always a guess due to the covered rectangular board.

However, the difficulties brought on by the first move cannot be avoided and are experienced by all solvers. Finding the best strategy for handling the first click will therefore be advantageous for all approaches. When a square is successfully probed, the player learns about the contents of the probed cell and the restrictions the cell places on its surrounding positions. For instance, the fact that cell x is a 4 indicates that x is not just what it is, but also that x's neighbors can only have a total of four mines. The player's goal is to maximize the amount of information revealed with each click.

However, when determining the best first step, every algorithm must take into account the trade-off between the possibility of getting a zero and the amount of information disclosed.

Another challenge is the handling of the guesses. In Minesweeper, being able to take deterministic actions is frequently insufficient for winning. Eventually, the player will be given too little information to draw any firm conclusions, so the possibility of performing deterministic actions in Minesweeper is often not enough to finish a game. The player receives too little information and therefore leaving them with little choice but will be forced to guess.

Without guessing, any Minesweeper solver could face issues for example like unable to finish certain configurations. Algorithms must decide on the same basis as the first click on whether to choose a corner, edge, or internal cell.

However, solvers must additionally think about the constraints that the investigated cells place on the unknown cells, the number of mines that are still in play, and the locations that will be most advantageous to the player.

The most naive approach to dealing with guesses is random selection. In random selection, any covered cell is selected and examined. The advantage of this approach is that it is easy to implement. Moreover, one can easily convince oneself that this method will never get stuck (no deadlocks). On the other hand, the main disadvantage of random selection is its simplicity. Random selection is not based on the knowledge that the player has about the state of the board.

However, in this project the random selection was used for guessing.

The most basic way of searching for something in a graph (such as a state with certain properties) is to check every node until the required node has been found. The most obvious algorithms for this are Breadth-First Search (uninformed search). BFS searches through the graph by first visiting the root node, then all the direct successors of the root node, then all the direct successors of those etc. The search visits all the nodes at a given depth before any deeper nodes. To avoid processing the same node more than once (as we are dealing with graphs, not trees, and possibly even cyclical ones) each explored node is stored and for all new nodes a check is performed against this storage. The time complexity of the algorithm is O(bd), where b is the branching factor of the graph (the number of successors each node has) and d is the depth of the solution. This requires that the nodes are tested for the termination criteria (i.e. whether the node is the one we are looking for) when generated rather than when expanded; in that case the complexity would be O(bd+1). Breadth-First Search is complete and optimal - that is, it is guaranteed to find the solution (given enough time and memory and assuming the solution depth and branching factor are finite) and the solution it finds is the lowest-cost one (if the cost of a path is a non-decreasing function of the solution depth, i.e. all the arcs in the graph have a non-negative cost associated with them). However, for many interesting problems the assumption about enough time and memory is not reasonable. With today's fast processors the main problem is memory - as the search needs to keep every generated node in memory. An algorithm that avoids this memory bottleneck is Depth-First Search. Instead of progressing all the way through each search depth before moving on to the next.

But because for this project the representation of any individual entity is not required and due to time issues, the mentioned algorithms above was not used for the project but instead propositional logic. Since propositional logic can either be true or false and never both it was the easiest way to implement the automated player for the minesweeper project.

## 3.3 Implementation & testing

For this mini project the task was to implement a "Tents and Trees" puzzle in Python.

The Tents (also known as "Tents and Trees") is a popular logic puzzle. According to https://wpcunofficial.miraheze.org/wiki/Tents, the puzzle was first published in 1989 by Léon Balmaekers under the name "Alen Verzamelen" in Dutch.

The rules of the puzzle are as follows:

* Find all of the tents in the forest.
* Every tent is attached to exactly one tree, and every tree is attached to exactly one tent.
* The number of tents is the same as the number of trees.
* The clues tell you how many tents are in that row or column.
* A tent can only be found horizontally or vertically next to a tree.
* Tents are never next to each other, neither vertically, horizontally, or diagonally.
* A tree might be next to two tents but is only connected to one.

**About Algorithm:**

I solved this problem using a step-by-step, iterative deepening search method (n= 10) because it can save some memory and is complete and optimal in comparison to other algorithms e.g. Breadth-first search.

Each cells of the grid may be 4 states : "tree", "tent", "grass", or "not-tested".

The steps of algorithm are as follows:

* Check all hints in rows and columns. If you find a row or column with a "0" hint, you can set all cells except for the trees in that row and column to "grass".
* And the rest of the steps are repeated until all tents are found.

1. Each identified "tree" and "tent" pair is excluded from the iteration. (Assume it is grass.)

Ein Bild, das Zelt, Baum enthält.

Automatisch generierte Beschreibung

1. Cells with no trees around them (top, bottom, left, right) are identified as "grass".

Ein Bild, das Pflanze enthält.

Automatisch generierte Beschreibung

1. If there is only one 'not-tested' cell around the 'tree', this is the 'tent'

Ein Bild, das Grün, Screenshot, Pflanze, Design enthält.

Automatisch generierte Beschreibung

1. All cells that exist around the cell identified as "tent" are set to "grass".
2. Check each hint and its rows and columns to determine the location and number of possible tents. Based on this information, make the identified cells into "tents".

Ein Bild, das Grün, Screenshot, Pflanze enthält.

Automatisch generierte Beschreibung

2

Ein Bild, das Grün, Screenshot enthält.

Automatisch generierte Beschreibung

2

1. Make the cells that exist around the new 'tent'(up, down, left, right) the 'grass'.

Also, if the current number of 'tents' in any row or column is equal to the hint, all remaining 'not-tested' cells become 'grass'.

1. See Hints and special location cases, cells that can't be 'tents' are set to 'grass'.

Like this:



1. check all the corners(meaning in subset) and "grass" the impossible ones.

Ein Bild, das Design enthält.

Automatisch generierte Beschreibung mit geringer Zuverlässigkeit

When you repeat this step, if the algorithm is satisfied with one step, it runs it again, starting from the first step. This is iterative method.

I don't think any one of breadth-first search, depth-first search, or others can solve this sensitive problem.

**User Interface**

The UI is built with streamlit. Streamlit is a powerful tool to create web UI with python for e.g. Artificial intelligence or data science.

**Requirements**

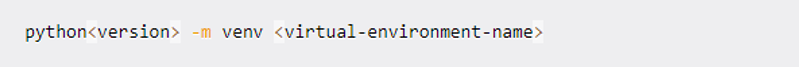
This app can run on Python 3.10.10. The latest version of Python is needed to run this program. Virtual environments are suitable for this app's installation package.

Follow these steps for python virtual environment installation (this is allow for both VS code or Pycharm):

1. Run this command in the terminal of VS code or Pycharm.



1. To use venv in your project, in your terminal, create a new project folder, cd to the project folder in your terminal, and run the following command:



For example:

Ein Bild, das Text, Schrift, weiß enthält.

Automatisch generierte Beschreibung

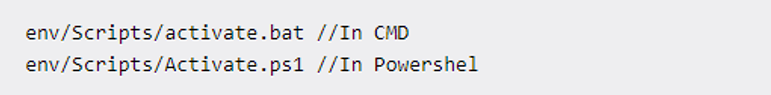
1. Now that you have created the virtual environment, you will need to activate it before you can use it in your project. On a mac, to activate your virtual environment, run the code below:

Ein Bild, das Text, Schrift, weiß, Screenshot enthält.

Automatisch generierte Beschreibung

This will activate your virtual environment. Immediately, you will notice that your terminal path includes env, signifying an activated virtual environment.

1. Note that to activate your virtual environment on Widows, you will need to run the following code below (See this link to fully understand the differences between platforms):



1. And you can install packages on virtual environment.

Run this:

Ein Bild, das Text, Schrift, weiß, Design enthält.

Automatisch generierte Beschreibung

Alternatively, if you can create a requirements.txt file, write the following command in the text and run it to install all the requirements packages for this app:



1. After package installation, you can run the app with this:

Ein Bild, das Schrift, Text, weiß, Kalligrafie enthält.

Automatisch generierte Beschreibung

For this project my app, [streamlit app name] is Streamlit\_app.py.

After that, you should see the app appear on your website.

**Structure of this app**

This project consists of four files : “Solver.py”, “main.py”, “Streamlit\_app.py” and “Images.py”.

* "Solver.py" contains all the functions in the algorithm that can perform all the steps in the algorithm.
* "main.py" processes the values from "solve.py" and outputs the results for step-by-step viewing in a terminal.
* “Streamlit\_app.py” which is required for the web UI.

Text output is following when running this app:

Ein Bild, das Text, Screenshot, Schrift, Grafikdesign enthält.

Automatisch generierte Beschreibung

Where "▲" stands for "tent" and "T" stands for "tree", "\_" for "uninspected", "□" for "grass" and "\_" for "untested" (since this is the last result, there is no '\_' to indicate 'untested').

The code was fully commented so in this chapter I will give just a short summary. In the first step, I searched for an proper search algorithm and then started with implementing the logic. I also added an “automated player” Button and tried to implement the minesweeper algorithm on it.

This project was also uploaded in my Github: DimitraBrou/Minesweeper\_Miniproject.

Like mentioned above, knowledge-based agents make decisions by considering their knowledge base and making inferences based on that knowledge. For this project the AI’s knowledge about a Minesweeper game is shown by making each cell a propositional variable that is true if the cell contains a mine, and false otherwise. So each sentence of the AI’s knowledge was presented {A, B, C, D, E, F, G, H} = 1 and every logical sentence in this representation was given two things: a set of cells on the board that are involved in the sentence, and a number count, representing the count of how many of those cells are mines. The above logical sentence says that out of cells A, B, C, D, E, F, G, and H, exactly 1 of them is a mine.

Deadlocks are game states that can never lead to a goal state. For this minesweeper project the implementation of a detection of a deadlock was not needed due the outcome of the game and the usage of randomness that cannot lead to a deadlock.

# 4. Consideration on ethics

“The essay should also contain the following additional consideration on ethics: Imagine that a variant of your solver will be deployed not in a computer game, but in reality. Think whether there are any military settings in which such a tool could be useful. Starting from such a military scenario, discuss the deployment of AI methods in military applications from an ethical perspective. Providing this discussion on ethics is required for passing.

One military application could be for instance, that such a deployment of AI methods in military applications would be used to send a drone into a minefield that has a metal detecting sensor that tells it, how many mines there are in the adjacent squares, while landing on a mined coordinate will cause a detonation. It could also be possible, to determine the amount of mines in the area beforehand by magnetic measurements.

To be able to give a better overview on an ethical perspective from starting with this scenario, it is necessary to understand in the first place how mines can be found in reality and what advantages it has, when the detection of a mine could be done via a drone with an algorithm over the air.

Identifying the precise locations of mines is one of the most important problems in landmine detection. Because of their unknowable locations and challenging detection, landmines are dangerous and therefore the removal require specialized knowledge. There are numerous methods for finding and removing landmines, as well as numerous difficulties that must be taken into account. Challenges could be changes in weather conditions that led to the change and disappearance of the landmines positons. Also, the landmines detection can get very expensive nowadays and every landmines detection technique has error rates. Therefore, the usage of a combined technique could decrease error rates. The current technologies for detecting a landmine are metal detector technologies, electromagnetic techniques, acoustic/seismic techniques, biological techniques, mechanical techniques and latest techniques (e.g. drone). And due to a combined technique via sensors on a drone, the risk of high error rates and the risk of humans getting hurt will decreases compared to manual detection. The risk of not being able to identify all mines could therefore also minimized (not like metal detectors that cannot differentiate a mine or from other objects).

In a nutshell, the development of an AI to detect mines could be very beneficial in risky landmines detection by using sensors and drones. Sensors collect precise data from the landmines field to discover anomalies by signals and image processing methods to detect mines.

But the effectiveness of any method should estimated by the deminers risks, false alarms and the success in mines detection rates. And because mines remain dangerous, a technology like an automated drone detector must continuously evolve to mitigate risks on its way to achieving optimal productivity and correctness. A technology like this have to be constantly monitored and ethical rules should be created and tracked during the development of such an technology and during the Usage of it. Also improved safety results from stricter regulations and processes and a stronger safety culture should be integrated and monitored.

Because using an autonomous drone to detect mines will guarantee the residents and deminers safety in one hand but on the other hand it could also be used for bad intensions. Also, there are some disadvantages of such automated technologies because AI’s cannot respond to emergencies and therefore this could cause inappropriate and wrong responses.

In order to list further disadvantages of the use of automatic drones, I would like to give first a look to the process of demining by drone. Currently, the area to be searched is first mapped, then the area is flown over with detector systems in order to detect mines, and then explosive devices are placed on the mine by means of a drone and blown up in a targeted manner. Now, however, such an algorithm could be developed for evil purposes, such as placing explosive devices by drone in places without mines or other hazards to cause more damage under the pretext of wanting to detect mines. Moreover, it can never be said how a self-learning AI will decide later and how it will develop, so it could make wrong decisions and place such explosive devices incorrectly.

While automated weapon systems are compatible with international law because humans retain command, this is not the case with autonomous weapon systems because there is no human control. This is because these autonomous weapon systems, from an ethical point of view, would lack human characteristics and important decisions would be made by them completely on their own without human influence. In other words, they would be adaptive systems that could act independently over a longer period of time without human influence. And they would be able to solve complex tasks and also to dynamically process new information resulting from changing influences and unpredictable developments and learn from it without human control. On this basis, they would be able to establish their own rules if necessary. It would therefore be impossible to predict how these systems would ultimately act. Humans would hardly be able to intervene any more. This would indeed be a new political, ethical and legal quality.

However, armed military drones could also be used as an instrument of warfare and cause a lot of damage. So if autonomous technical systems are developed, in which the "human factor" no longer plays a role, the ethical question arises as to who can be held responsible for war crimes in the event of misconduct by fully autonomous drones.

To sum it up, there are countless possibilities of useful AI methods in military applications that could keep away dangerous situations from the human being. In our case it would be possible to detect the coordinates of mines and even causing the detonation of them to avoid injury of military personnel. Despite the obvious benefits of deploying AI methods in military applications in terms of function we need to preconceive that no AI is impeccable. Considering the enormous amount of possible dangerous situations that could occur in military applications caused by defective AI the price to pay is invaluable. Every incorrect assumption could eventually end not only one but many human lives. Therefore, the right choice of the algorithm and how the corresponding algorithm is implemented is important for the development phase. Afterwards, the AI should be tested to prevent errors and misbehavior in time. For example, it would save a lot of resources such as money, drones, etc., because if the algorithm misbehaves, the drone could be stopped before it is damaged or explosives could be placed in the wrong places.

From my ethical perspective it is useful to deploy AI methods in military applications to reduce risk for human beings as long as the development is censoriously verified and the maintaining of the AI method is always guaranteed and monitored. With that not being guaranteed I would clearly not rely on AI itself.

Also, an artificial intelligence can only ever be as smart as it is appropriately designed to be. In addition, it should always be taken into account that the use of artificial intelligences can eliminate jobs and in the future, more university graduates could be hired instead of people with manual professions, as these could be replaced. In addition, people who interacting with AI systems must be able to retain full and effective self-determination over themselves and participate in the democratic process.

Therefore, when implementing such an AI, it would be important to ensure that humans can intervene at any time. Because in real life it can happen for various reasons that not the shortest way is needed. So, each AI could implement a kind of emergency brake in the form of an "exit button". This exit button should be used so that in case of disagreement by the human user, the human can intervene in the decision of the AI and prevent certain decisions. So that the final control remains with the human, and he can make his life decisions himself, because it is not possible to implement an AI which only considers ethical decisions.

Another important aspect would be the data protection and data management that could be regularly controlled by independent third parties in the form of audits and there could be a data protection officer on site for each AI area. In addition, regular audits could also be conducted for compliance with technical robustness, cybersecurity, and compliance with fundamental rights and code of ethics, which could verify required standards, policies, fundamental rights, standards requirements, transparency, or legal requirements. So, because algorithms for example like the breadth-first search algorithm, needs a lot of memory and space, the data management is an important point.

Also, before implementing an AI, a benefit analysis should be done by multiple parties such as the government, agencies, stakeholders, and the end users. The AI should not be developed until it can generate benefits for all parties, and no one is disadvantaged, and no party lacks security.

A domain-specific code of ethics should absolutely be developed prior to the development of an AI. The development of an AI should then be developed based on it. The code and fundamental rights should be regularly updated, and tests could be created to ensure that the code of ethics is still being used in the AI. For example, differnet testings could be done regularly to see how the AI would act and react in certain situations and whether those actions still fit with the code of ethics.

1. Own illustration [↑](#footnote-ref-1)