**Intro to AI I Project**

1. **Intro and Rules**

“Tents and Trees” is a fun puzzle created by Léon Balmaekers with the Dutch name "Alle Ballen Verzamelen" in 1989. It is not too dissimilar from Sudoku and Wumpus World, both of which we have covered in some form during the course.

The objective of the game is to place tents on a rectangular grid of varying dimensions. The grid is pre-filled with an arbitrary number of trees in a combination of positions and the rules are established for how and where tents can be placed. While the core of the game is always the same, some minor differences or “unwritten rules” can be found by trying different implementations online. As far as this project is concerned, the rules were interpreted exactly as written on the [Moodle page](https://moodle.aau.at/course/view.php?id=38447) for the course and are as follows:

* Place tents on the grid, so that each tree has a unique horizontally or vertically adjacent tent. This also means that no tent can "serve" more than one tree.
* Tents should not be adjacent to each other (neither horizontally, nor vertically, and also not diagonally).
* The given numbers for rows and columns must be equal to the number of placed tents in the respective row or column.

*A note:* Throughout this document and in the code for my solver, I will refer to “horizontally or vertically adjacent” as **bordering** whereas the term **adjacent** itself will be considered to include diagonals as well.

1. **Considerations**

While these rules are sufficient to provide a general solver, some cases are left ambiguous. In particular, while a well-defined problem always allows for a unique solution, it is not clear from the specifics of the project whether an input could be given that represents an ill-defined problem, in which case there are a few scenarios to consider:

1. The problem does not allow a solution. This could be either because the sums of the numbers for rows or columns (i.e. the total number of tents to place) don’t match up, are lower than the number of trees, or because it is not possible to find a configuration of tents that simultaneously serve all trees without being adjacent to one another.
2. The problem allows for multiple solutions. This might happen if the sum of the numbers for rows or columns is higher than the number of trees and there is enough room on the board to place tents in multiple non-adjacent ways. This is made possible by the “strict” interpretation of the given rules that do not specify that tents should be bordering trees, only the other way around. Therefore, according to this interpretation, it’s possible for a tent to be fully isolated without any trees and obviously other tents in the surrounding cells.
3. The problem definition contains a formal error. Technically speaking, this could apply to 1. as well but I wish to make a distinction for clarity. I consider part of this category problems in which the input is otherwise corrupted (non-rectangular grids for example).
4. **Approach**

Having made these considerations, the design process required choosing the best approach to tackle the problem itself and the edge cases that we wish to take into account. First off, I pondered the edge cases.  
I think it goes without saying that a good solver should not only be able to find the correct solution when one exists, but also confidently identify problems for which no solution is possible. Accordingly, it seemed a no-brainer to include scenario **1.** in my solution.  
The decision is not as clear-cut when it comes to scenario **2.** Since this scenario arises from an ambiguity left by the conciseness of the rules, it is arguable whether it should be worthy of inclusion. In my case, I figured that adhering to the rules as proposed as strictly as possible was the best approach and consequently decided to allow for this case to exist.  
Finally, when it comes to case **3.** I decided that formal errors should be considered as oversights on behalf of the user and the responsibility to identify them should not befall on the solver. In such cases the behaviour of the solver is undefined, it may find a solution if one exists or not, depending on the error.

After the conceptual groundwork has been established, comes the time to think about the actual implementation of the logic in terms of languages, tools and techniques. The most obvious path to me is to use **Answer Set Programming** (**ASP**) to define the constraints and provide the facts that make up the input topology. To speed up the development process, I decided to use the ASP-Chef Recipe provided on Moodle to parse the input problem and output the facts and I developed the code for the solver in an online interactive environment to be able to quickly test and iterate over versions of the program. The environment is provided by [potassco.org](http://potassco.org) and is based on the command-line tool **Clingo**.

The **ASP** code for the solver is made available in the attachments and online on [GitHub](https://github.com/SKRYMr/TentsAndTrees). The code is extensively commented to provide a clear explanation of every clause. It should run on any publicly available Clingo version, however the versions used during development and the only versions tested are **Clingo 5.7.0** and **Clingo 5.7.1**.

After the initial development stage had concluded, I moved the code into a **Python** script that would allow me to parse the input files and output a solution with a single command. I used the standard library **argparse** to collect user input from the console and the publicly available **clingo** library to run the **ASP** program, again published by potassco. The **ASP** program itself is saved as a *.txt* file in the same directory as the Python script.

The **Python** version used is **3.11.7.**

To test the solver, problems were sourced from <https://www.puzzle-tents.com/>, painstakingly converted to the input format by hand, fed into the Python script and double checked on the same website by translating the solution by hand. Approximately 20 problems with different levels of complexity and different grid sizes were tested this way after development. Some more were used during development.  
In addition, a few examples for each of the edge cases discussed above were conjured by me and hand-fed into the solver to check for its behaviour.

1. **Ethics Discussion**

While this puzzle can provide a stimulating past-time, there are other similar settings in which the same techniques could be employed to find solutions to more practical problems.

Among those are many military settings in which strategical placement is crucial to operation success that would benefit from employing the same solver.  
We can think of a defensive situation in which we are tasked to place camps and bases near sensitive targets for quick response. We want the bases to be near the potential targets and to be directly connected to them via major infrastructure (an abstraction of the bordering rule). We don’t want any sensitive target to be left untended and we cannot risk a single base tending two targets at the same time lest we fail to respond to a double coordinated attack. Finally, we wouldn’t want our bases to be too close to one another for multiple reasons: A single powerful strike (e.g. a nuclear attack) could take out more than one at the same time; if enemies managed to capture one base, they would be in an advantageous position to capture the adjacent ones; we have a limited amount of forces at our disposal and it is optimal to spread them around to be able to ideally cover more ground.  
Swap trees with targets and tents with camps and we already have a solution for this situation.

So far, this example has been fairly neutral from a moral standpoint and could be even considered positive if we traded military camps for red cross stations and hospitals ready to provide relief to sensitive targets in an emergency situation. I want to propose an alternative application that showcases the reverse, where the same solver could be applied with malicious intent in a military setting.

Suppose we are trying to protect our military structures and relevant targets from shelling in an urban environment under siege from enemy fire. Knowing that international conventions impose laws (that too often have been forgotten or overlooked in recent times) to govern conflict that forbid the targeting of essential civilian welfare structures like hospitals and culturally or historically significant buildings like churches, museums and monuments, we could strategically place temporary military infrastructure as close as possible to these off-limits targets to make enemy decision making harder and easier to condemn in front of a global audience. While this would make for a valid and efficient strategy, it is undoubtedly morally corrupt to put civilian lives and heritage at risk to further war efforts.

These two scenarios perfectly illustrate how technology has no moral compass of its own and its apport to humanity is fully dependent on the people who use it. More advanced artificial intelligence has the peculiar ability to often elude the understanding of even the very same people who made it. For this reason, deploying artificial intelligence in military settings requires thoughtful considerations on the governing principles behind its development and extensive testing to ensure the explainability of its decisions. In summary, there are two key components to responsible innovation and usage of technology:

* Virtuous development, aimed at contributing positively to the lives of people and tasked with solving problems with an anthropocentric vision. In general, it’s fundamental to establish and communicate clearly the goals and objectives of any development project, stating the means that will be required to achieve such goals and why the developers think it’s a good idea to do so, in essence what is their vision. Ideally, developers should also be aware of potential bad actors, how they might exploit their product to nefarious ends and what might be done to prevent it. Especially when it comes to artificial intelligence, it’s vital to ensure that the biases and shortcomings present in data gathered from human experience are identified, isolated and acted upon and that the model is stable and tested enough to guarantee that the outputs it produces are in line with the vision of the developers with a satisfying margin of confidence. This does not mean that the developers should take on full responsibility for the actions of the users nor that they should limit the capabilities of the software in a way that hurts the usability in the name of ethics.
* Informed use, done conscientiously and with explicit intent by the end user. It is too easy nowadays to follow the narrative of machines taking over the world and acting of their own volition against humankind but in truth almost all cases of harm coming to individuals or categories from technology are rooted in malice or human error. Some of the most infamous examples are deepfakes, especially in connection to pornographic content, and the ever-present “machines are going to steal our job” argument. Both these issues arise from an erroneous handling of the power of new technologies and could be solved through socio-political action and information campaigns. For an ethical integration of artificial intelligence in our society it is necessary that everyone does their part and people start to take responsibility for their actions in an ever more connected world.

When it comes to military applications in particular, both principles must coalesce to ensure that the technology is meant to simplify conflict by shortening its length and reducing the number of casualties on all sides, especially civilians, and to minimize collateral damage to the countries’ infrastructures, structures and environment. Also, the people responsible for its use must be properly trained and act with integrity in accordance with the established laws and guidelines, must be aware of how the technology works and the risks it entails and must be prepared to be held accountable for anything that goes wrong.