Project : Generating and Solving 3D Tetracube Puzzles with Answer Set Programming

Nicolas Melaerts
Umons, Faculté des Sciences, Belgique

Kenza Khemar
Umons, Faculté des Sciences, Belgique

Abstract

This report presents an approach based on Answer Set Programming (ASP) to solve the problem of placing tetracubes in a 3D volume. We will explore how to model and solve the problem of placing 8 different tetracubes (pieces composed of 4 unit cubes) in a volume of 32 unit cubes. We present three configurations: a 2x4x4 rectangular prism and a 2x2x8 rectangular prism and 2 regular prism of 2x2x4. Our solution will use the Clingo solver and includes an interactive 3D visualization in python of the solutions found. We will analyze the performance and discuss the challenges encountered in modeling this combinatorial problem.

2012 ACM Subject Classification Theory of computation → Constraint satisfaction

Keywords and phrases Answer Set Programming, Tetracubes, 3D Puzzle, Clingo, Combinatorics

1 Introduction

Puzzles involving the placement of pieces in a given volume constitute a classic challenge in recreational mathematics and computer science. In this report, we focus on the specific problem of placing tetracubes in a 3D volume.

A tetracube is a geometric figure composed of 4 unit cubes connected face-to-face. There are 8 different types of tetracubes: I, T, L, Pyramid, O, N, Z, and Z-mirror. Our goal is to place these 8 tetracubes in a volume of 32 unit cubes, without overlap and completely filling the space.

We explore three volume configurations:

- A rectangular prism of dimensions 2x4x4
- A rectangular prism of dimensions 2x2x8
- 2 regular prism of 2x2x4

To solve this problem, we use Answer Set Programming (ASP), a declarative programming paradigm particularly suited for combinatorial and constraint satisfaction problems.

2 Problem Modeling in ASP

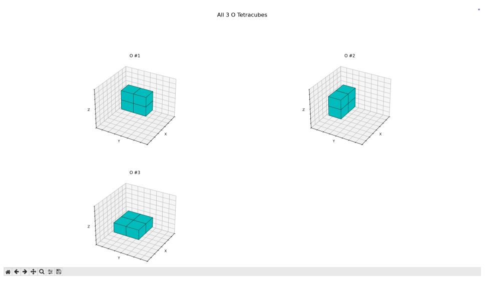
2.1 Tetracube Representation

Each tetracube is represented by a set of 4 unit cubes positioned in 3D space. For each type of tetracube, we define all its possible orientations (rotations). For example, the I tetracube can be oriented in 3 different directions (aligned with the X, Y, or Z axis).

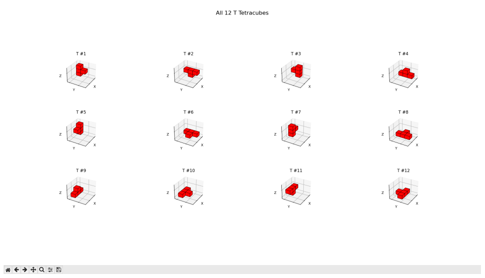
The number of unique orientations varies significantly between tetracube types. The O tetracube, with its compact shape, has only 3 unique orientations, while the T tetracube has 12 different orientations due to its asymmetric structure. This variation in the number of orientations affects the complexity of the placement problem.

- The I tetracube has 3 unique orientations
- The L tetracube has 24 unique orientations

- The N tetracube has 12 unique orientations
- The O tetracube has 3 unique orientations
- The Pyramid tetracube has 10 unique orientations
- The T tetracube has 12 unique orientations
- The Z tetracube has 12 unique orientations
- The Z-mirror tetracube has 12 unique orientations



(a) The O tetracube with its 3 unique orientations



(b) The T tetracube its 12 unique orientations

Figure 1 Examples of tetracubes with their possible orientations

2.2 How to represent tetracubes in ASP

In our ASP model, each tetracube is represented as a collection of 4 unit cubes positioned in a relative coordinate system. We define a predicate `cube(Type, Rotation, DX, DY, DZ)` where:

- `Type` is the tetracube type (I, T, L, Pyramid, O, N, Z, or Z_mirror)
- `Rotation` is a unique identifier for each possible orientation
- `DX`, `DY`, `DZ` are the relative coordinates of each unit cube

For example, the O tetracube (shaped like a 2x2x1 rectangle) has 3 unique orientations, corresponding to its alignment with each of the three axes:

Listing 1 ASP representation of the O tetracube

```
% O tetracube
% Rotation 1
cube("O",1,0,0,0). cube("O",1,0,0,1). cube("O",1,0,1,0). cube("O",1,0,1,1).
% Rotation 2
cube("O",2,0,0,0). cube("O",2,0,0,1). cube("O",2,1,0,0). cube("O",2,1,0,1).
% Rotation 3
cube("O",3,0,0,0). cube("O",3,0,1,0). cube("O",3,1,0,0). cube("O",3,1,1,0).
```

Each rotation is defined by specifying the coordinates of the 4 unit cubes that make up the tetracube. The first rotation places the O tetracube in the XY plane, the second in the XZ plane, and the third in the YZ plane.

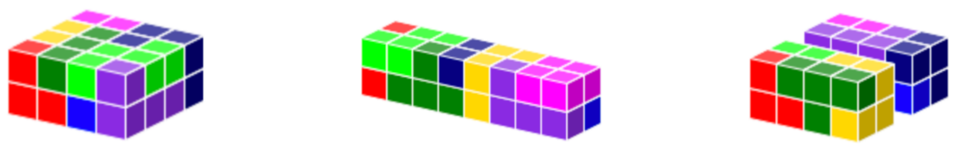
More complex tetracubes like the L shape have many more possible orientations (24) in total, due to their asymmetric structure. Each orientation must be explicitly defined in our ASP model to allow the solver to consider all possible placements.

This representation allows us to efficiently model the placement of tetracubes in 3D space while ensuring that all geometric constraints are properly enforced.

2.3 Problem Definition

Our ASP model will defines:

- The 3D grid (2x4x4, 2x2x8 or two 2x2x4)
- The 8 types of tetracubes and their possible rotations
- Constraints ensuring that:
  - Each tetracube is placed exactly once
  - Tetracubes do not overlap
  - All unit cubes in the grid are occupied
  - Tetracubes remain within the grid boundaries



(a) 2x4x4 configuration **(b)** 4x2x8 configuration **(c)** Two 2x2x4 configurations

Figure 2 Different volume configurations for tetracube placement. Source: `https://puzzler.sourceforge.net/docs/polycubes.html#tetracubes`

2.4 Puzzle Generation

To generate engaging puzzles from our model, we will introduce additional constraints that create partially solved configurations. The idea will be to find valid complete solutions first, then remove some pieces to create puzzles of varying difficulty:

- We can fix the positions of one or two tetracubes as "clues" and challenge the player to place the remaining pieces
- Different difficulty levels can be created by varying the number and type of pre-placed pieces
- A progressive puzzle could involve completing the structure level by level, with each level adding complexity

This approach will transform our solver into a puzzle generator. By analyzing the solution space, we will identify configurations that are challenging but solvable, creating an engaging 3D puzzle experience.

2.5 ASP Encoding

Here, we present what the main parts of our ASP encoding should look like:

- Grid definition
- Representation of tetracubes and their rotations
- Placement rules and constraints
- Puzzle generation

3 Implementation

3.1 Project Structure

Our project will be organised as follows:

- `tetracubes.lp`: Definition of the 8 tetracubes and all their rotations
- `BIG_CUBE.lp`: ASP model for the 2x4x4 cube
- `BIG_CUBE_2.lp`: ASP model for the 2x2x8 cube
- `TWO_SMALL_CUBES.lp`: ASP model for the two 2x2x4 cubes
- `PUZZLE.lp`: ASP model for the puzzle generation include the 2x4x4, 2x2x8 or two 2x2x4 cubes with some pieces fixed
- `place_tetracubes.py`: Interactive 3D visualization of solutions
- `draw_tetracubes.py`: Helper functions for visualization

3.2 Solving with Clingo

We will use Clingo, a modern ASP solver, to find solutions to our problem. Before attempting to visualize or generate a puzzle, we will first need to check whether a stable model exists that is, whether a valid solution can be found for the given constraints and grid configuration.

If Clingo returns a stable model, then we will proceed to use that solution as a basis for puzzle generation or visualization.

```
clingo BIG_CUBE.lp -n 1
```

And for the 2x2x8 cube:

```
clingo BIG_CUBE_2.lp -n 1
```

And for the two 2x2x4 cubes:

```
clingo TWO_SMALL_CUBES.lp -n 1
```

Once a stable model (i.e., a valid solution) is found, we can proceed to generate a puzzle from it. This is done using the PUZZLE.lp file, which takes the solution as a base and removes some of the pieces to create a partially filled configuration.

We can pass as a parameter the number of tetracubes that should remain fixed as clues. For example, the following command will generate a puzzle where 3 tetracubes are pre-placed:

```
clingo BIG_CUBE.lp PUZZLE.lp -c fixed=3 -n 1
```

This allows us to control the difficulty of the puzzle by adjusting the number of pre-placed pieces. The fewer the fixed pieces, the harder the puzzle becomes. This turns the solver into a flexible puzzle generator, capable of producing unique and solvable 3D challenges.

3.3 Solution Visualization

We will develope an interactive 3D visualization tool in Python using Matplotlib. This tool allows:

- Displaying the solution in 3D
- Rotating and zooming on the model
- Progressively adding/removing pieces to understand the assembly
- Distinguishing different types of tetracubes by colors

4 Results and Analysis

4.1 Solutions Found

Here we will present the solutions found for the different configurations.

4.2 Performance

Here we will analyze the performance of our approach.

- Solving time for each configuration with a definite seed
- Number of models (solutions) found

5 Conclusion

In this project, we will develop an Answer Set Programming approach to solve the challenging problem of placing 8 different tetracubes in a 3D volume of 32 unit cubes. Our implementation will demonstrate the effectiveness of ASP for modeling and solving complex combinatorial problems with geometric constraints.

We plan to explore three different volume configurations (2x4x4, 2x2x8, and two 2x2x4 prisms) and create a flexible puzzle generation system that can produce challenges of varying difficulty by fixing different numbers of pieces. A key aspect of our work will be not just solving the placement problem, but also generating engaging puzzles that can be presented to human solvers with varying levels of complexity. The interactive 3D visualization tool we will develop will enhance understanding of the solutions and make the puzzle more accessible.

This work will showcase how declarative programming paradigms like ASP can elegantly handle complex spatial reasoning tasks. Future work could explore extensions to other volume shapes, additional geometric constraints, or optimization techniques to improve solving performance for larger instances of similar puzzles.

A Complete Source Code

Here we will present our source code.

A.1 ASP Model for the 2x4x4 Cube

Listing 2 BIG_CUBE.lp

```
% ASP model definition for the 2x4x4 cube
% ...
```

A.2 ASP Model for the 2x2x8 Cube

Listing 3 BIG_CUBE_2.lp

```
% ASP model definition for the 2x2x8 cube
% ...
```

A.3 ASP Model for the Two 2x2x4 Cubes

Listing 4 TWO_SMALL_CUBES.lp

```
% ASP model definition for the two 2x2x4 cubes
% ...
```

A.4 ASP Model for the Puzzle Generation

Listing 5 PUZZLE.lp

```
% ASP model definition for puzzle generation
% ...
```