CSCE625: Artificial Intelligence
Programming Assignment 5: Packing Puzzle

Dylan Shell

This is posted the week of March 10, 2022. The intended submission date is April 6, 2022. Submission details are on the third page.

Question

This assignment involves implementing a search procedure for a simple geometric problem where local search can be effective.

The tetrominoes are shapes made from four squares (here 'tetra' meaning four, '-ominoes' from dominoes). There are five basic shapes and you are likely familiar with them from the classic game tetris. Like tetris, we will consider shifts and rotations (by multiples of $90°$); unlike tetris, we also allow the inclusion of a *reflection*. Figure 1 shows the five basic shapes: **T**, **I**, and **O** are invariant under reflection. While **J**/**L** and **S**/**Z**) are not.

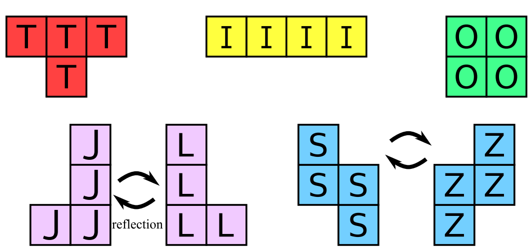

Figure 1: The set of tetrominoes has five pieces made of 4 blocks. In this assignment we consider the tetrominoes to be 'free' to that translations, rotations by multiples of $90°$, and reflections are all allowed.

One complete collection of tetrominoes fills $5 \times 4 = 20$ units. Given some rectangle of height h and length ℓ, such that area $h \times \ell = 20 \cdot k$ for some $k \in \mathbb{N}$, is there a way to pack exactly k collections of tetrominoes into the rectangle?

Task

In a language of your own choosing, implement code that given ℓ and h will attempt to answer this question in the affirmative by seeking a suitable packing.

We are interested in a method which is fast and effective, so it is not required that your code be complete, i.e., that it determine and report that no packing exists.

Submission

Generate a sequence of queries of increasing size (with differing aspect ratios), and fix a reasonable timeout (say ± 5 minutes), and then run your code on each query for that timeout.

Write a report showing a few examples of your code being run and the solutions it finds. Your report should be no more than a couple of pages of the main content, but that should include:

- Your name and UIN.
- Notes that can help explain the algorithm and approach taken so that it is easy to understand your code.
- If you did something especially cunning, or had a clever idea you wish to share, document this fact.
- Specific notes about known bugs, issues, limitations, or errors. If you had some parameters or items to tune (e.g., a GA with a population size, or SA with a cooling schedule) provide some description of what you tried in order to obtain your final version.
- Documentation of resources used and/or help received.
- Affix your code as an appendix. (Not counted toward the page quota.)

Submission of the document (as a PDF) will be facilitated via the canvas site. The deadline posted on the course webpage will be the official date.