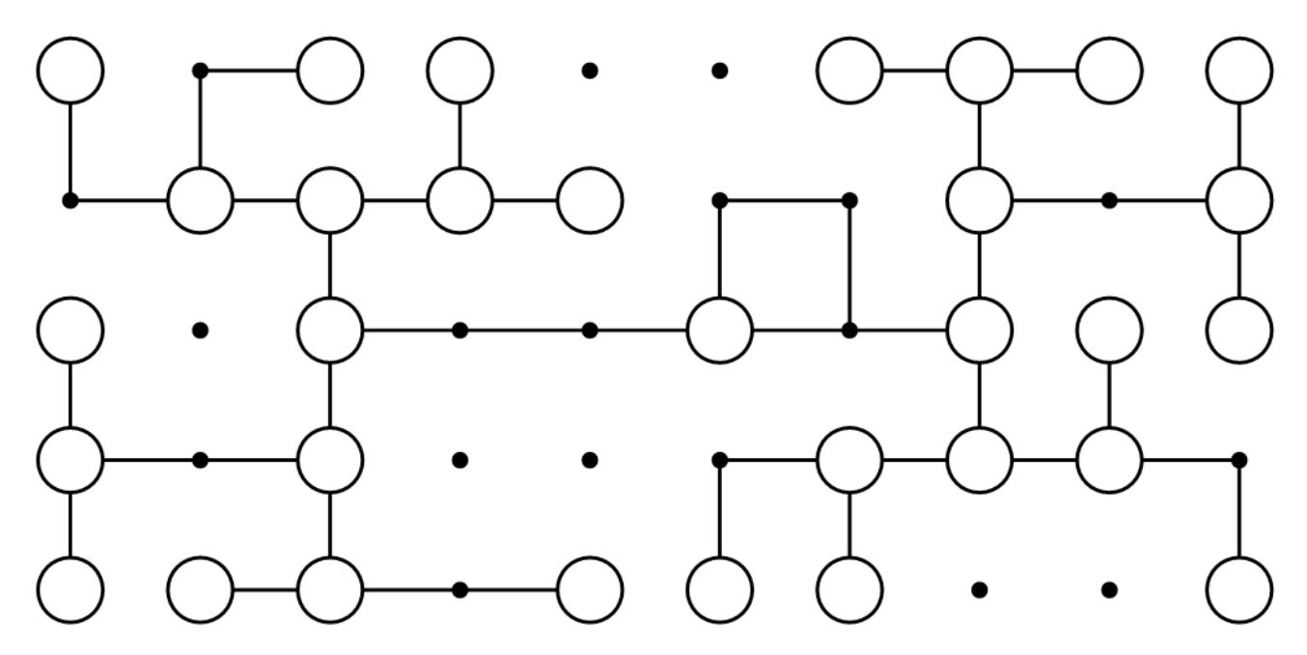
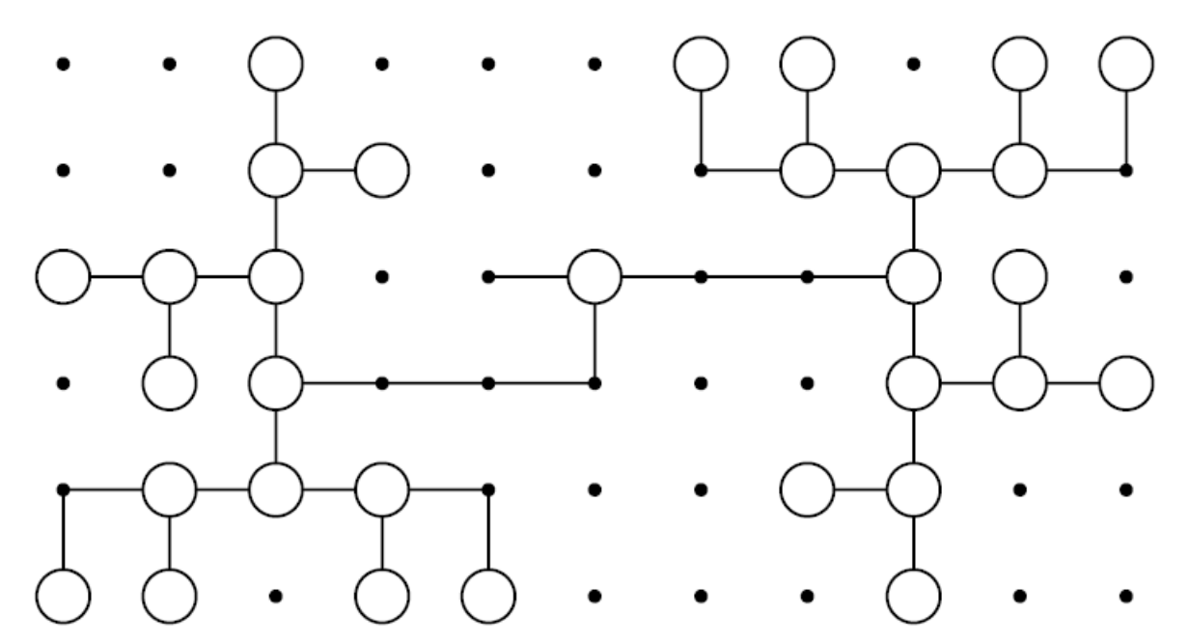
Embedding Binary Trees into Grids Using Genetic Algorithms

John Granville
University of Puget Sound



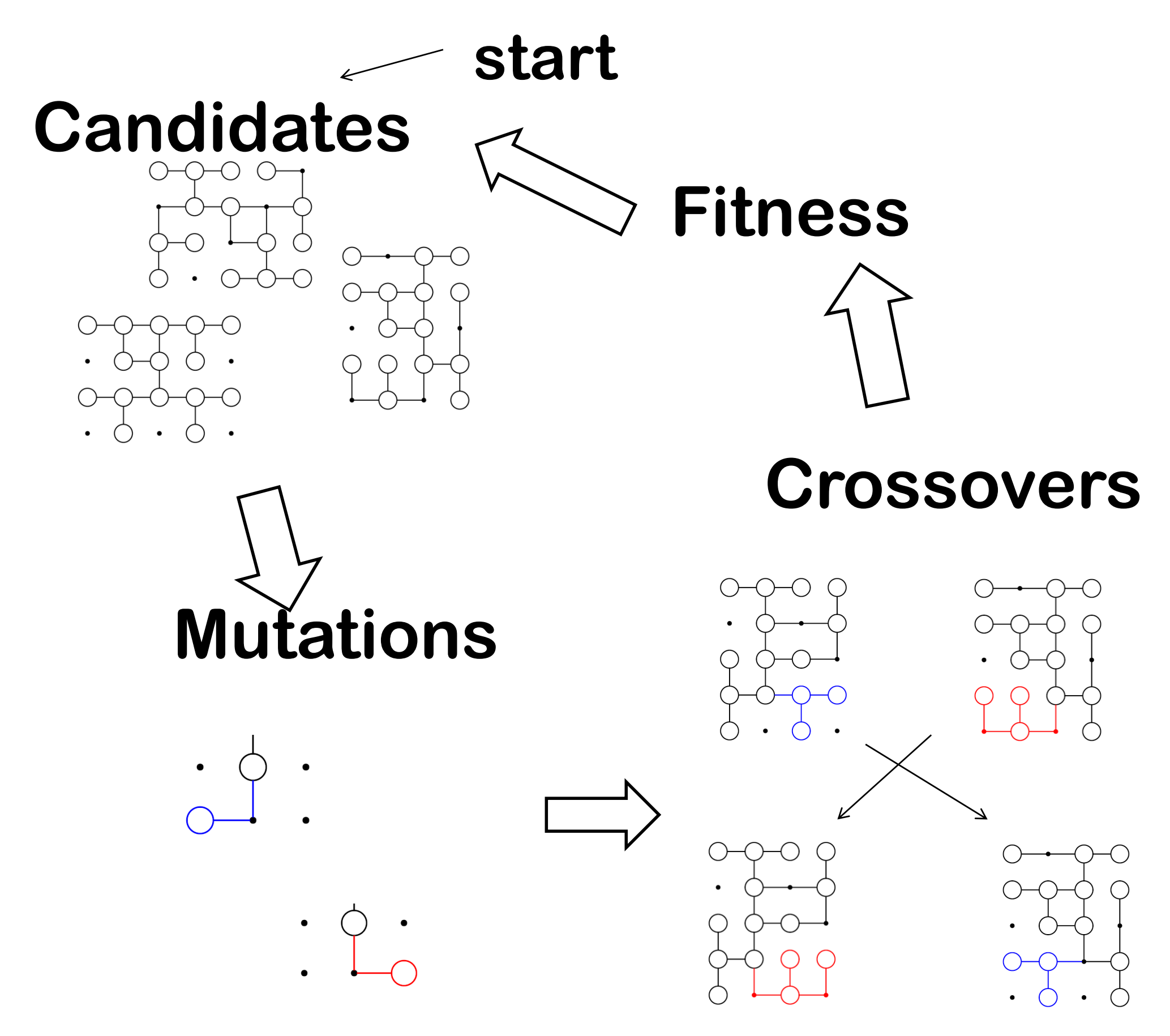
Embedding Binary Trees In Grids Using Genetic Algorithms



Joe Granville, Department of Mathematics and Computer Science, University of Puget Sound. Advisor: Brad Richards

Genetic Algorithms

Genetic algorithms are a problem-solving method based on evolution: a pool of candidate solutions are evaluated for fitness, the best of which are mutated and/or bred to create the next generation of candidates.

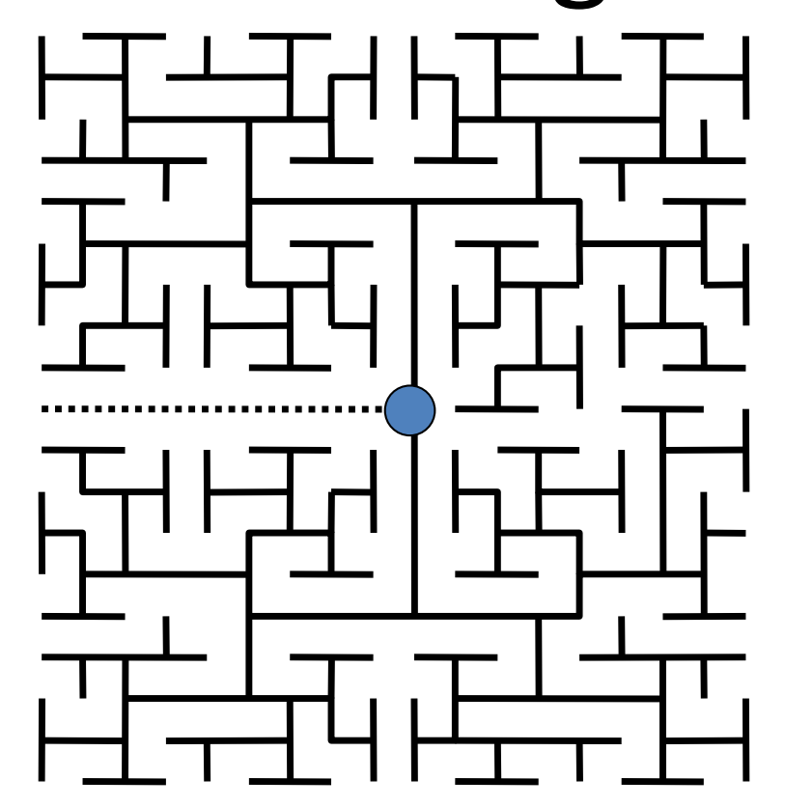


Efficiency

The efficiency of embeddings is the primary factor in ranking the candidates, but their fitness is also affected by the amount of overlap present. Efficiency is given by the ratio of the size of the grid needed to contain the mapping of the tree and the number of nodes in the tree.

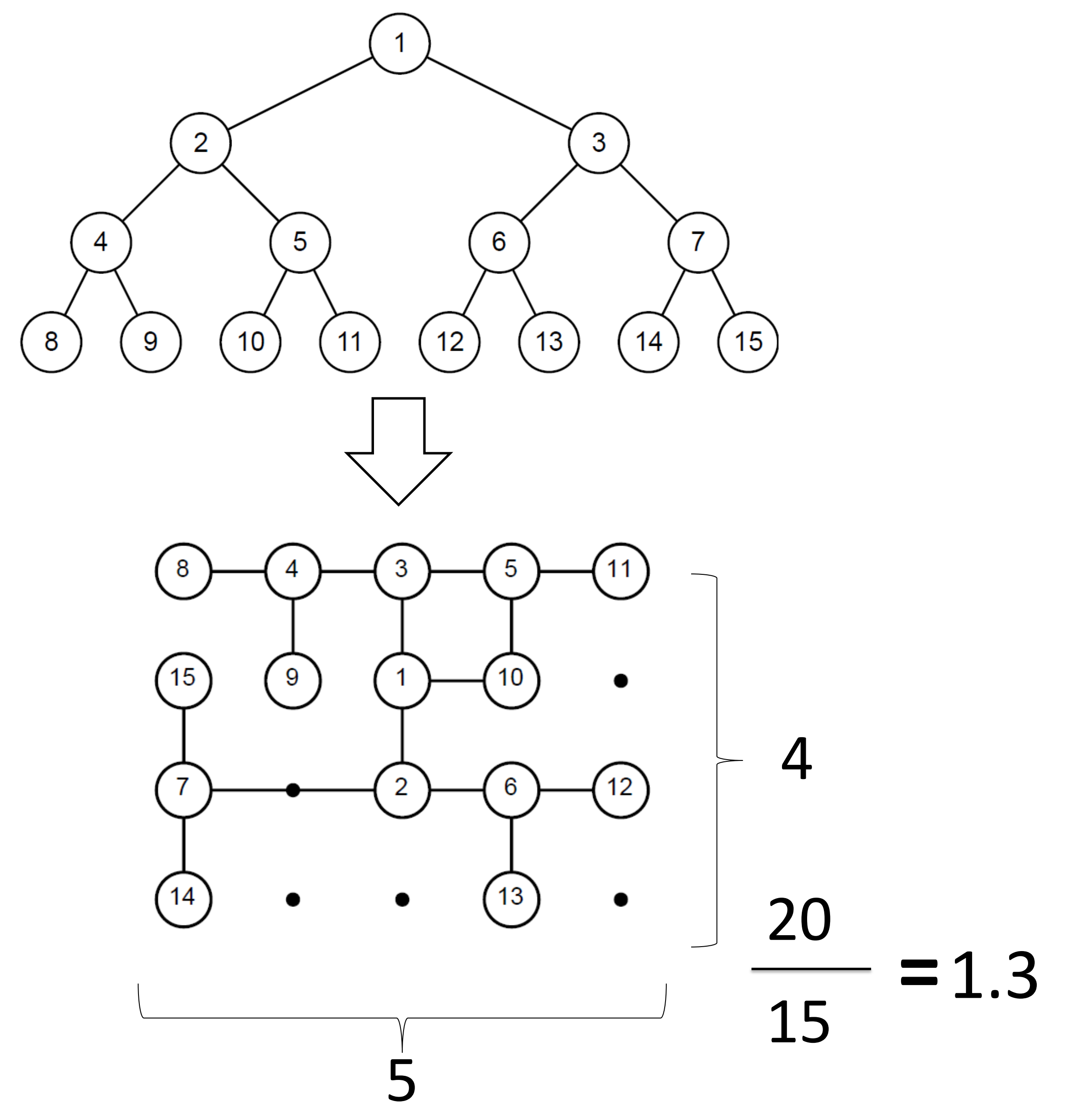
Previous Work

Lin, et. al. were able to attain efficiencies as low as 1.3 (as seen in the embedding pictured below) by using small hand-made embeddings and combining them. Stephanie Hatley worked on a computational approach to the problem, but it did not allow any overlap at any point. As a result her program could not create embeddings for trees larger than height three.



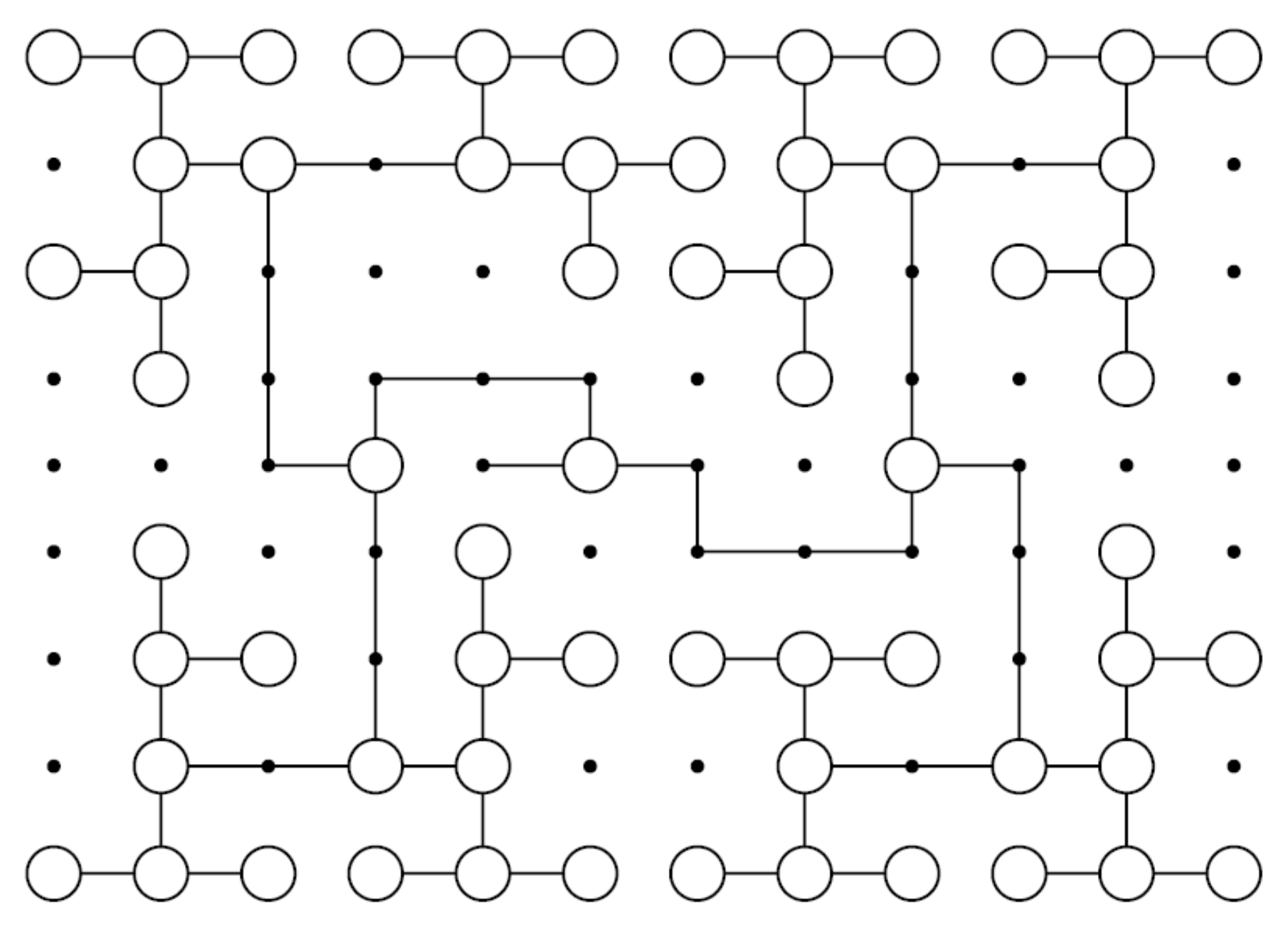
Problem Description

I developed a genetic algorithm to create efficient embeddings of binary trees into grids computationally. Creating an embedding involves placing the nodes of the tree on adjacent vertices of a grid, or vertices that can be connected by a path. Overlap, created by a path going through a node or another connection, is not allowed in the result. Embeddings can be used in parallel computing as well as in creating layouts for integrated circuits.



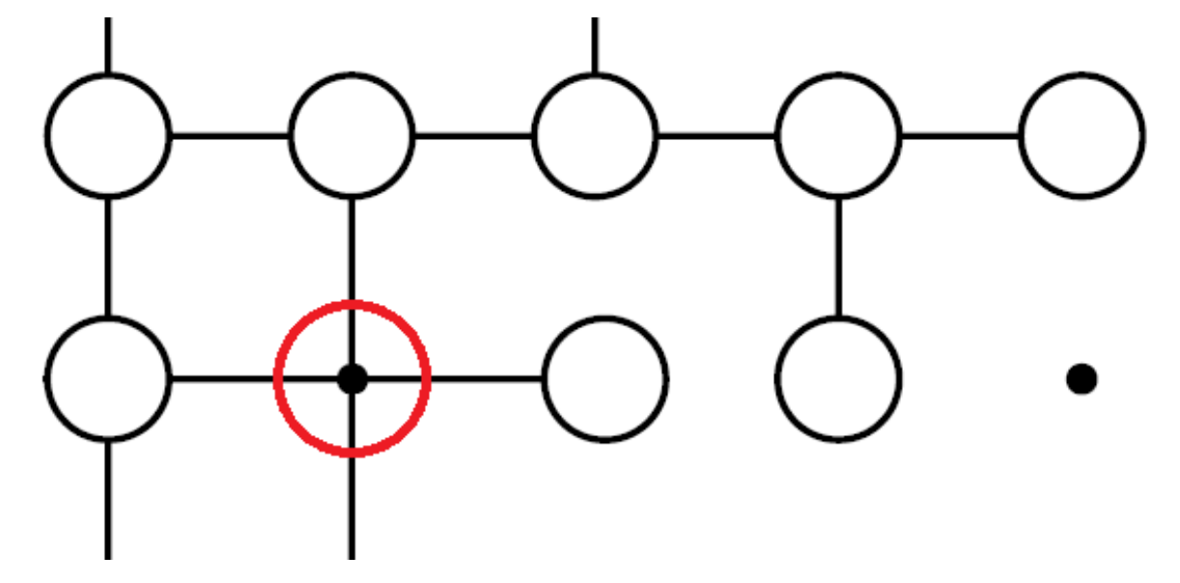
Results

By tolerating overlap and stitching smaller trees together my program has been able to create tree embeddings up to height 5 with efficiency lower than 2. The tree embedding pictured below has an efficiency of 1.79, and is the best height-5 tree my program created. At height 4, I can create embeddings as efficient as 1.4.

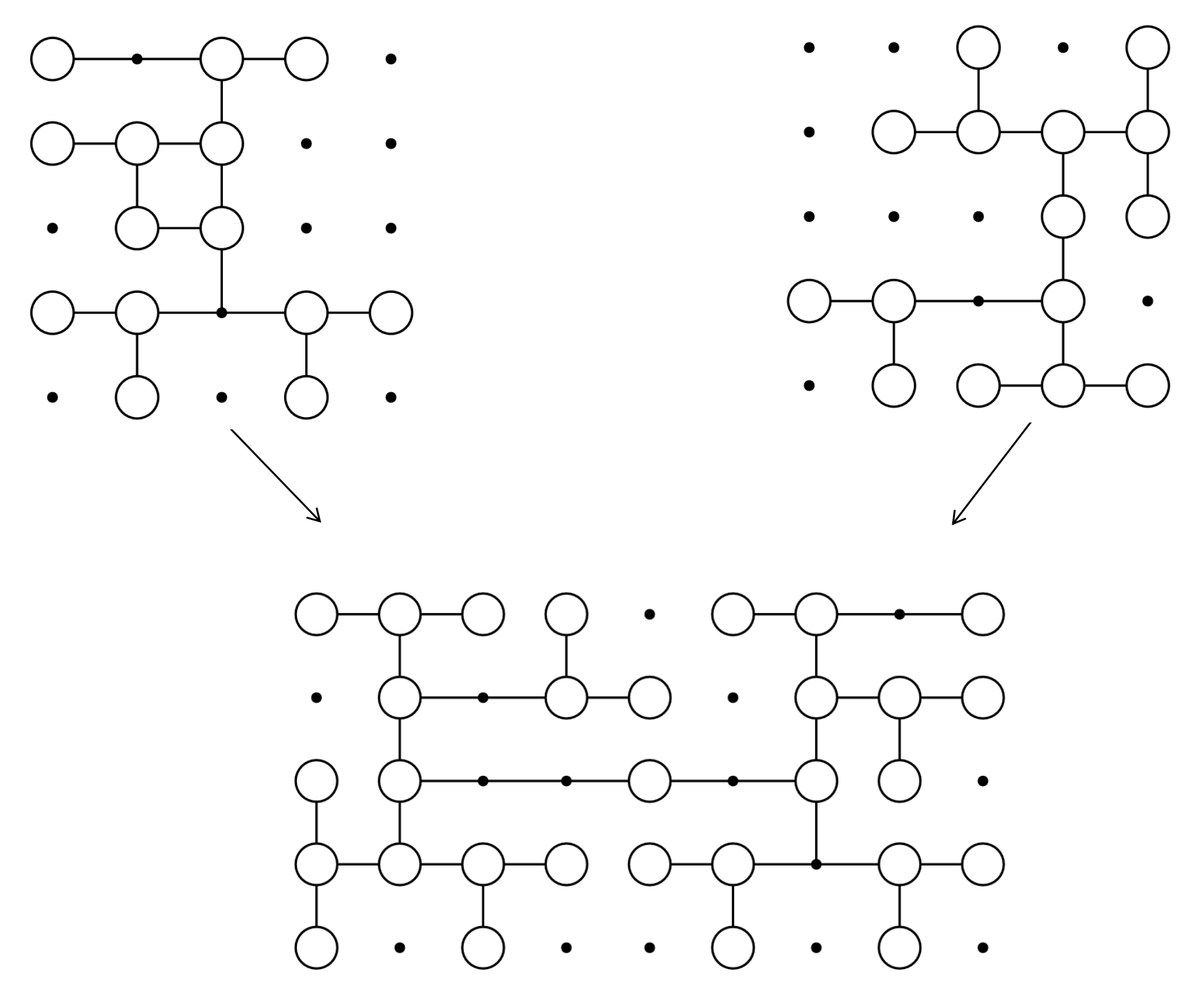


Penalty Functions

On larger trees, random mutations and crossovers frequently produce overlap (seen below), which is not allowed in the final result. However, temporarily allowing overlap can lead to highly efficient embeddings. I added two methods to tolerate but penalize overlap: one gives a penalty based on how many overlaps there are in the entire population, while the other increases the severity of the penalty as the program runs. The two methods working in tandem allow overlap early on while limiting it in later stages.



Stitching: The path to larger trees



In order to create more efficient embeddings of larger trees, I developed a technique in which two smaller trees are stitched together. This technique can be applied recursively to reach trees of any height. While I was able to make height-3 tree embeddings with efficiencies below two, anything bigger than that was inefficient without stitching, ranging from 10 to as high as 92.

Future Work

- Refine stitching function
- Further work on overlap tolerance functions
- Stitching with more than two trees

References: Lin, Y.B., et. al. *Expansion of Layouts of Binary Trees into Grids* , Journal of Discrete Applied Mathematics, 2003, 611-642