Introduction
We strive to develop an algorithm that can find the solutions to a given Shirokuro puzzle in a reasonable amount of time. We employ a simple yet powerful technique known as “backtracking,” which incrementally builds candidate solutions and abandons each partial candidate (backtracks) as soon as it determines that the candidate cannot possibly be extended to create a valid solution.

Backtracking essentially amounts to a depth first search through the tree of all partial candidate solutions to a problem. Each time a partial candidate solution is determined to be a dead end, a portion of the tree is pruned, resulting in a smaller search space. Good backtracking algorithms can efficiently detect dead ends at nodes that are close to the root, so that the pruned subtrees are as large as possible.

Dead End Detection
Below are two nontrivial Shirokuro configurations that our backtracking algorithm currently detects as dead ends.

This above configuration, known as the diamond, leads to the separation of at least 1 pair of disks. To see this, note that any connection that is made between the two black disks invariably forms a loop which completely encapsulates one of the white disks, making any connection between the two white disks impossible.

If two black disks lie along the boundary of the grid, they must be connected by a chain of black disks, all of which also lie along the boundary. To see why, consider the alternative: suppose a pair of black disks and a pair of white disks are arranged along the boundary so that such a chain would be impossible, as illustrated above (left). Then any connection drawn between the two black disks would bisect the grid, isolating the pair of white discs from one another (right).

Detecting Isolated Groups
Since no colored disk may become isolated from others of it’s kind, the early and efficient detection of isolated groups of disks is essential for achieving a reasonable runtime. To this end, we have developed a technique which involves building and maintaining a pair of spanning trees.

$S$ is the current state of the Shirokuro solving process. $S_{\text{white}}$ and $S_{\text{black}}$ are modified versions of $S$ formed by filling every empty cell with white or black disks, respectively. The tree shown in $S_{\text{white}}$ spans the region that the white disks in $S$ may potentially connect with.

$S_{\text{white}}$ and $S_{\text{black}}$ are modified versions of $S$ formed by filling every empty cell with white or black disks, respectively. The tree shown in $S_{\text{white}}$ spans the region that the white disks in $S$ may potentially connect with.

An advantage of using spanning trees is the ease with which they can be restructured. In figure above, the spanning tree connecting the black disks is broken into 3 pieces when one of it’s nodes (circled) is removed. The smaller pieces are searched and found to contain nodes that can be reconnected (starred). These nodes become the new roots of their respective subtrees in the final configuration.

Dynamic Variable Ordering
In most backtracking algorithms, the order in which variables are instantiated is not fixed, but instead changes dynamically by following several variable ordering heuristics. One important heuristic, the degree heuristic, states that those variables that are involved in a larger number of constraints should be instantiated first.

In our future work, we will be exploring the role that our spanning trees may have in dynamic variable ordering.