Joshua Cords

Dr. Tang

CS 420

11 November 2016

Local Search – Project 2 Report

**Requirements**

Use two Local Search algorithms to solve 19-Queens problem. The first Local Search is Steepest-Ascent Hill Climbing; the second is the Min-Conflicts algorithm. The algorithms will be compared by generating one thousand random starting positions of 19-Queens and finding the percent of solved cases. Because the Min-Conflicts algorithm also requires a max-moves parameter, parallel results will be produced for max moves or 50, 75, 100, 125, 150, and 175. Steepest-Ascent Hill Climbing is run in parallel for each max-moves parameter, allowing a baseline to keep the random starting positions from skewing the results.

Input User selects number of queens, number if iterations, and the max-moves for the Min-Conflicts algorithm.

Output For each algorithm, outputs the number and percentage of final solutions at each number of attacking queens. The output also includes the average duration of each solution recorded in microseconds.

**Method**

Implemented Steepest-Ascent Hill Climbing and Min-Conflicts algorithm according to pseudocode. Used a single dimension array size n to track each state, each array element represented a column and the number stored in each array element represented the row of the queen’s position in that column. Created a custom check to count number of attacks optimized to check the minimum number of squares. Attacks along a row, column, or diagonal where counted as follows, 2 queens threatening is 1 attack, 3 queens is 3 attacks, and so on. Queens are allowed to see through other queens to determine the number of attacks.

A timer was used to keep track of the time elapsed, via System.nanoTime.

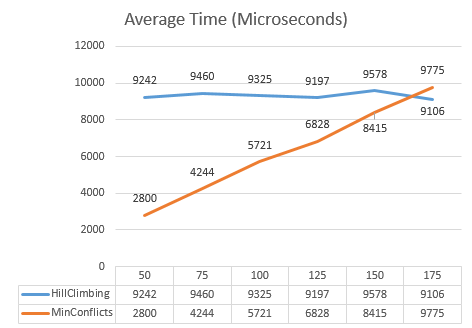
Steepest-Ascent Hill Climbing considers each possible move of all queens and chooses the next position based on the best result. Min-Conflicts only considers one random column at a time and chooses the next position based on the best move that one queen can make.

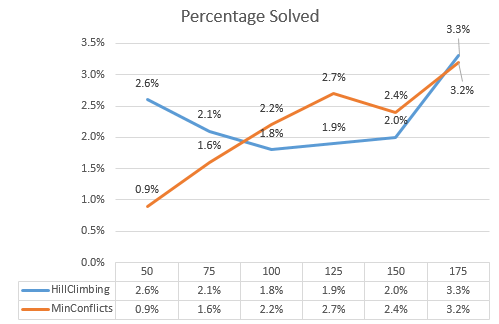
**Testing**

Each algorithm was tested for 1000 random starting positions for each max-moves parameter: 50, 75, 100, 125, 150, and 175.

**Findings**

Here are the output and results from the Local Search algorithms.





**Analysis**

As the max-moves increased to and beyond 100, the Min-Conflicts algorithm outperformed the Steepest-Ascent Hill Climbing except at 175. At 50 and 75 max-moves it appears Min-Conflicts did not have enough moves to solve a position all the way through to completion, and the lower percentage is likely due to positions that would have been solved given another 25 to 50 moves.

As max-moves increased, time also increased until the point where max-moves was set to 175 and Steepest-Ascent Hill Climbing outperformed Min-Conflicts. Therefore, if time to completion is the secondary goal after percentage solved, it is important to pick a max-moves that high enough to solve possible positions to completion, but low enough to satisfy the time constraint.

It was surprising to find that Min-Conflicts generally outperformed Steepest-Ascent Hill Climbing when max-moves was set above 75. This suggests that making the best possible move from each position creates on average more dead-ends, positions that cannot be solved to completion, than randomly choosing a column to improve.