Course Project: Stochastic Local Search for Variable Ordering in Graphical Models

For my project I implemented a novel Stochastic Local Search approach for finding a good variable elimination ordering. A good variable elimination ordering is defined as one with low induced width. I compared my implementation with Dr Kalev Kask’s min-fill heuristic, and Vibhav Gogate and Rina Dechter’s QuickBB algorithm. The latter is introduced in the following paper:


Pseudo-code for the SLS algorithm:

**Induced-Width-SLS**

**Input:** An undirected graph $G = \{X, E\}$, $X = \{x_1 \ldots x_n\}$, $E = \{e_1 \ldots e_m\}$

**Outputs:** An ordering $d$ of $X$, the induced width along this ordering: $w^*(d)$

**Initialize** $d$ to a min-fill ordering of $X$

**do**

if $\text{Rand} = \{0..1\} < 0.05$

  go to a random adjacent ordering

else

  go to an adjacent ordering $d'$ that greedily improves $w^*$
  if no such ordering exists go to a random adjacent ordering

**until** elapsed time == time cutoff

**return** $d'$, $w^*(d')$

**Claim:** The complexity of Induced-Width-SLS can be bounded by $O(n^2)$

**Proof:** at worst, Induced-Width-SLS will check all $n$ adjacent orderings. The worst-case complexity of calculating the induced width of a given ordering $d$ is $O(n)$. It follows that Induced-Width-SLS can be bounded by $O(n^2)$.

**Definition:** Two orderings $d$ and $d'$ are adjacent if and only if $d=<X_1\ldots X_n>$ and $d'$ differs from $d$ by at most two variables such that: $<\ldots X_i, X_{i+1}\ldots>$ in $d$ appear as $<\ldots X_{i+1}, X_i\ldots>$ in $d'$

It follows from this definition that the size of the neighborhood = the number of adjacent orderings = $n - 1 = O(n)$
However it is the author’s suspicion that the average-case run-time is $O(nd)$ where $d$ is the max of the degrees of the two nodes involved in the swap. Limited empirical analysis supports this claim (omitted)

**Results** *please see attached slides*

The following graphs were taken from Professor Bodlaender’s Treewidth Library
<http://people.cs.uu.nl/hansb/treewidthlib/index.php>. These are accepted treewidth benchmarks: Instance #1, #2

The following graphs are random partial k-trees: #3 is a partial 60-tree, #4 is a partial 100-tree, #5 is a partial 200-tree
each of these instances is guaranteed to have $w^* \leq k$
however due to my randomization scheme, the induced width of these graphs are likely less than k.

**Conclusions**

Induced-width-SLS is beat by a good implementation of min-fill. There was no instance where Induced-width-SLS provided a better ordering than min-fill. However the quality of the ordering is highly dependant on the initialized ordering. Initializing Induced-width-SLS with the best ordering found by min-fill may find orderings better than using either technique alone.

**Future Work**

In addition to experimenting with different design decisions, I am rewriting the code in C. My intention is to optimize the code to get each iteration to run as fast as possible.