LAB 3

Kumar Ayush - 2015eeb1060

Course Instructor - **Dr. Narayanan C. Krishnan**

Indian Institute of Technology, Ropar

1. Kakuro Solver using CSP

How to run

Use *g++ -std=c++11 <filename.cpp>* to run the code. Input file name could be changed inside the code itself(at line #9). Output will be printed in *output.txt* (could be also changed in the code at line #10). Node and Arc consistency could be enabled/disabled at line #15 and line #16.

Method

Implemented in C++. Backtracking has been implemented using recursion and is basically a DFS. Node consistency has been implemented using external memory to keep track of domains. Similarly for MAC, external memory was used to properly manipulate domains.

Observation

Input size: 5 x 6

Backtracking:

Time elapsed: 9e-06 seconds

Backtracking calls: 16

Virtual memory: 56464; Resident set size: 41800

Node Consistency:

Time elapsed: 3.1e-05 seconds

Backtracking calls: 16

Virtual memory: 56464; Resident set size: 42280

Arc Consistency:

Time elapsed: 7.5e-05 seconds

Backtracking calls: 16

Virtual memory: 56464; Resident set size: 42240

Input size: 10 x 10

Backtracking:

Time elapsed: 51.5699 seconds

Backtracking calls: 34864933

Virtual memory: 56464; Resident set size: 42304

Node Consistency:

Time elapsed: 76.2624 seconds

Backtracking calls: 35875019

Virtual memory: 56464; Resident set size: 42176

Arc Consistency:

Time elapsed: 215.27 seconds

Backtracking calls: 35875019

Virtual memory: 56464; Resident set size: 42228

Input size: 14 x 12

Backtracking:

Time elapsed: 215.27 seconds

Backtracking calls: 35875019

Virtual memory: 56464; Resident set size: 42228

Node Consistency:

Time elapsed: 38.023 seconds

Backtracking calls: 26531832

Virtual memory: 56464; Resident set size: 42508

Arc Consistency:

Time elapsed: 166.177 seconds

Backtracking calls: 26531832

Virtual memory: 56464; Resident set size: 42508

We observe that for relatively lower input size, simple backtracking outperformed MAC and node consistency. It's because of the nature of the algorithm implemented. For maintaining domain values in MAC, extra set of memory and time is utilized which dominates the total running time without MAC. That's why we see slower performance in MAC and traditional backtrack. The time and memory for backtrack is a lot less than for tracking domain values. The number of backtracking calls are relatively same.

For moderately sized inputs, similar case is observed. The memory use is almost same. Time taken is significantly high in MAC. Number of backtracking calls are also slightly higher.

For relatively larger inputs, the number of backtracking calls get reduced in MAC and node consistency. The time taken is still quite high for MAC. Memory utilization is almost same.

Above observation shows that though Arc consistency may reduce the number of backtracking calls, it needs to be optimized to be effective. My algorithm is quite poorly optimized and thus the running time with MAC is significantly slower. Few data structures could be used to significantly improve running time, for example, priority queues. Also, various heuristics like least constraining value and minimum remaining values could significantly impact running time.

2. Kakuro Solver using MiniSAT

Method

Not implemented