Rajul Ramchandani & Conan Zhang

Professor Thomas Henderson

CS 4300 - 001

22 September 2016

Assignment A3: Arc Consistency Algorithms

1. INTRODUCTION: CONAN

Constraint Satisfaction Problems are situations where a set of objects must match a number of given limitations. A solution is a given by a state where all objects satisfy the constraints of the problem. Arc Consistency is a form of checking local constraints of two objects where one object's domain is checked against another object's domain to ensure that they both meet the given limitations. This can then be used to find solutions to a given Constraint Satisfaction Problem. We will be analyzing different implementations of Arc Consistency Algorithms to solve the N-queens Problem. The N-queens Problem is a puzzle where you must place N queens on an NxN board so that none of them threaten each other. Arc Consistency can solve this as the local constraints for each queen can be checked to ensure a solution is possible.

The two implementations of Arc Consistency that we will be looking at are the AC-1 and AC-3 algorithms. The questions we seek to answer are:

1. What are the number of ones before and after the application of our constraint algorithms?
2. What are the execution times for each algorithm given a different N and probability of 1s?
3. What are the given conditional expected reductions for different Ns and a given starting number of 1s?
4. What are the best regression fits and constants of the complexity functions for the two algorithms?

2. METHOD: RAJUL

The method for this experiment was to create a program that can run a large number of trials on 2 different algorithms to simulate solving the N-queens problem. This is done by timing the two algorithms and plotting it on a graph while comparing it to existing functions to find its complexity. The two algorithms we use, make use of a matrix structure that essentially denotes the board for the queens. In the matrix, each column represents each queen (as they need to be in different columns) and the columns would be the domains labels which are essentially the

position. Each possible position is left as a 1, where the algorithm for the revise method changes these 1s to 0s if the queen cannot be placed at that situation. The two Algorithms are AC-1 and AC-3. Each case involved number of queens from 4 to 10, a probability of 1s in the D matrix, and 200 trials for each case.

Both Algorithms use two methods: Revise and a P function that checks restraints. The revise function is basically the algorithm that goes through the given arc and uses P to implement the restraints. It does this by making changes to the D matrix turning 1s to 0s where necessary, making sure that the current positions has support. P is usually the method that defines what this "support" means.

Since both Algorithms use these methods, what is different is that AC-3 takes advantage of a queue that consists of all arcs. What is unique about it is that the algorithm keeps a collection of arcs that are yet to be checked. When the domain of a variable has any values removed by revise(), all the arcs of constraints pointing to that pruned variable are added.

Whereas, AC-1 almost brute forces the same, going through all arcs. It does this multiple times until a full pass over the matrix shows no change.

3. VERIFICATION OF PROGRAM: CONAN

To verify that our program runs correctly, we will be testing our implemented AC-1 and AC-3 algorithms against 4 know results for preset D matrices.

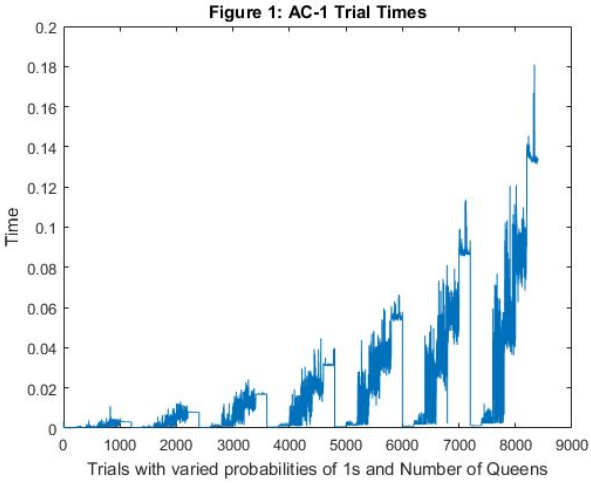
D1 Matrix				**D1 Known Solution**				**D1 AC-1 Solution**				**D1 AC-3 Solution**			
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

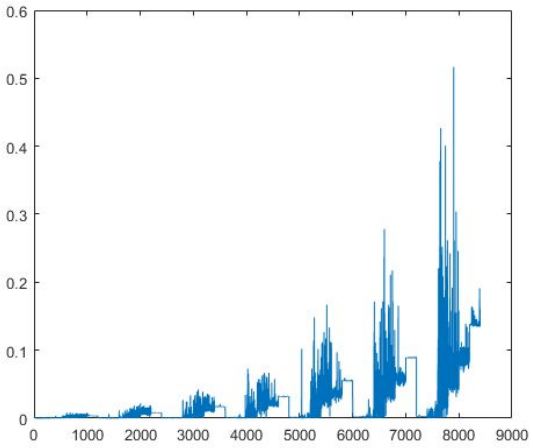
D2 Matrix				**D2 Known Solution**				**D2 AC-1 Solution**				**D2 AC-3 Solution**			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

D3 Matrix				D3 Known Solution				D3 AC-1 Solution				D3 AC-3 Solution			
0	1	0	1	0	1	0	0	0	1	0	0	0	1	0	0
1	1	1	1	0	0	0	1	0	0	0	1	0	0	0	1
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
1	1	1	1	0	0	1	0	0	0	1	0	0	0	1	0

D4 Matrix				D4 Known Solution				D4 AC-1 Solution				D4 AC-3 Solution			
1	1	1	1	0	0	1	0	0	0	1	0	0	0	1	0
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
1	1	1	1	0	0	0	1	0	0	0	1	0	0	0	1
1	1	1	1	0	1	0	0	0	1	0	0	0	1	0	0

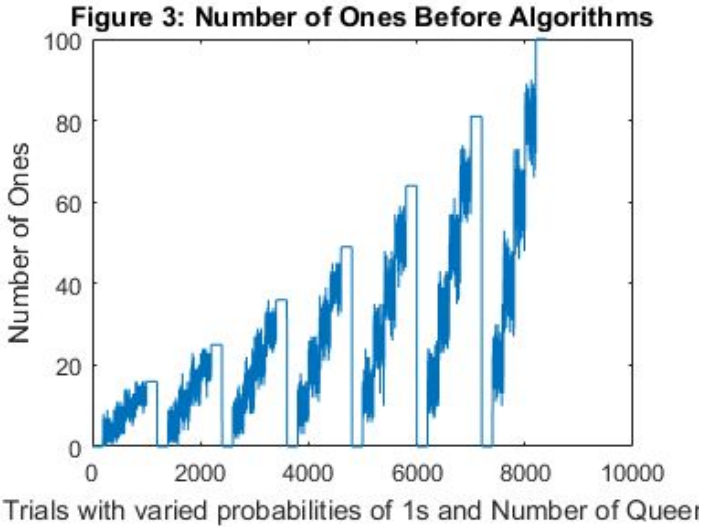
4. DATA AND ANALYSIS: RAJUL

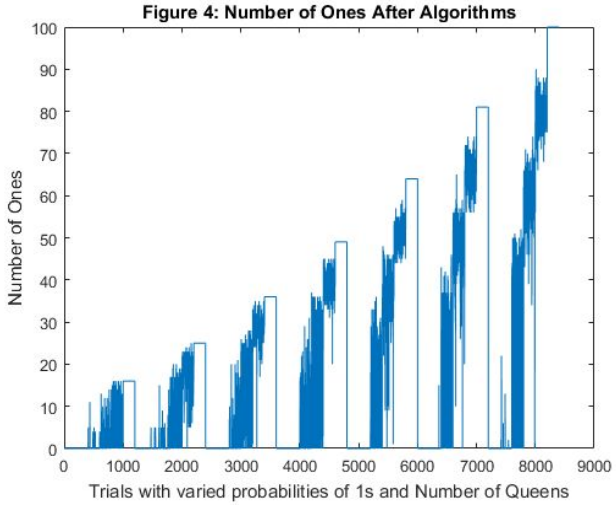




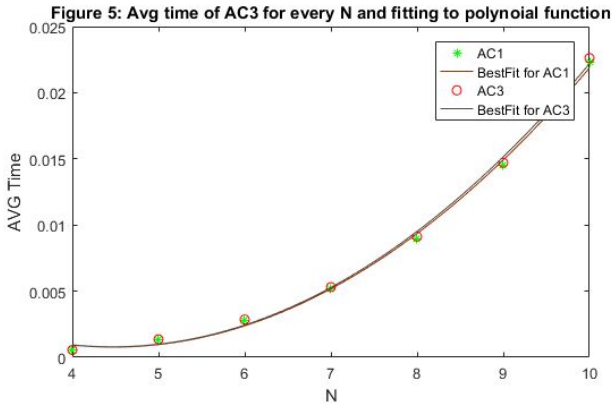
The graphs above show the different times for AC1 (Figure 1 Left) and AC3 (Figure 2 Right). As it is observed below, each graph contains 7 sets of data, each being for N = 4:10. For ease of comparison the above graphs were left as plots of all 200 trials. From the above, AC3 get pretty large when compared to AC1 when compared at the second half of the graph. To analyze more closely, it seems that AC3 is faster for till N = 8 (rough;y) and some cases for N=9 as well. At large cases - N=9 and N=10 and AC1 seems to be faster.

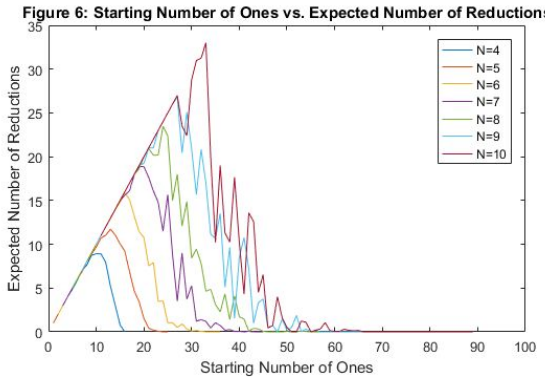
Below, are the graphs for number of 1s before and after the CSP is applied. Here , it is observed that the number of ones after the CSP algorithm is applied is less than the number of ones before. This is due to the fact that that the algorithm turns 1s to 0s to try and solve it. In the graphs below (Fig 3 and 4) for every case of N, there are more values close to the X axis hence denoting that there are less 1s.





The above graph is the time complexity graph of AC1 and AC3 while varying N. First, comparing the best fit lines, the best fit line for AC1 is above (very close) to the line for AC3 in the first half before N=7. Then, it clearly drops below the line for AC3 denoting that it was slightly faster. Comparing the points plotted for time taken, while there





is a slight variation between the points, it is difficult to tell due to the scale of the graph. The points should follow the trends of the bestfit lines as well.

Figure 6 provides the conditional expected reductions for different Ns for a given starting number of 1s.

5. INTERPRETATION: CONAN

Summarize your results and explain them in simple physical terms whenever possible. Specific questions that were raised in the assignment should be addressed here. Also, give suggestions for future work or possible extensions. You need to answer (directly) questioned posed in Section 1.

From the above data, the number of ones understandably decreases after the CSP is applied. Although there are spikes at points where a lot of ones still remain in the unsolved matrix, overall there is a decrease as expected.

As for the times plotted in Figure 1 , Figure 2 and Figure 5 it can be concluded that the implementation of AC1 and AC3 used in this experiment followed a trend. With all these graphs, it can be concluded that the time taken by AC3 is usually faster towards lower N values roughly around N=8. After this value, AC1 seems to be slightly faster contradicting our earlier belief that AC3 is always faster. Figure 5 shows the complexity of the functions as well as the best fit. The best fit functions were made with constants obtained with the time data.

Additionally, Figure 6 provides the conditional expected reductions for different Ns for a given starting number of 1s. As N increases the possible number of 1s understandably also goes up do to the increase in size of matrix.

6. CRITIQUE: RAJUL

From the above, we did see that AC3 did seem like it was relatively slower than AC1. This may be due to the fact that best timing practices were not followed and overhead of creating large matrices.

While writing the code, it was very important to consider the corner cases needed for AC1 and AC3. Since the matrices were expanding and were larger than 16 elements, it was hard to verify the larger matrices to see if the solution given was entirely correct. This posed an experimental issue of even being able to verify the code for AC1 and AC3 for higher N values. In addition, we learned that the use of priority queues and adding neighbors for AC3 definitely was supposed to make it faster, while AC1 was a more primitive brute force style technique.

From the above, we did see that although AC3 was supposed to be much faster, but towards bigger matrices AC1 seems to become faster. This seems contradictory but at the same time could be defended by the fact that AC3 goes through a lot of comparisons adding neighbors and so has many comparisons. Unfortunately, as mentioned above, this is difficult to verify. Either a robust version of the AC algorithms or correct verification of timing practices will definitely help the data be more accurate.

7. LOG: RAJUL & CONAN

Keep a log of the time spent on each problem and include it in the report.

Rajul: I spent 5 hours on AC1, 6 hours on AC3, and 7 hours on the Lab Report.

Conan: I spent 5 hours on AC1, 6 hours on AC3, and 7 hours on the Lab Report.