CPSC 322 2018W2 Assignment 2 solutions

Question 1:

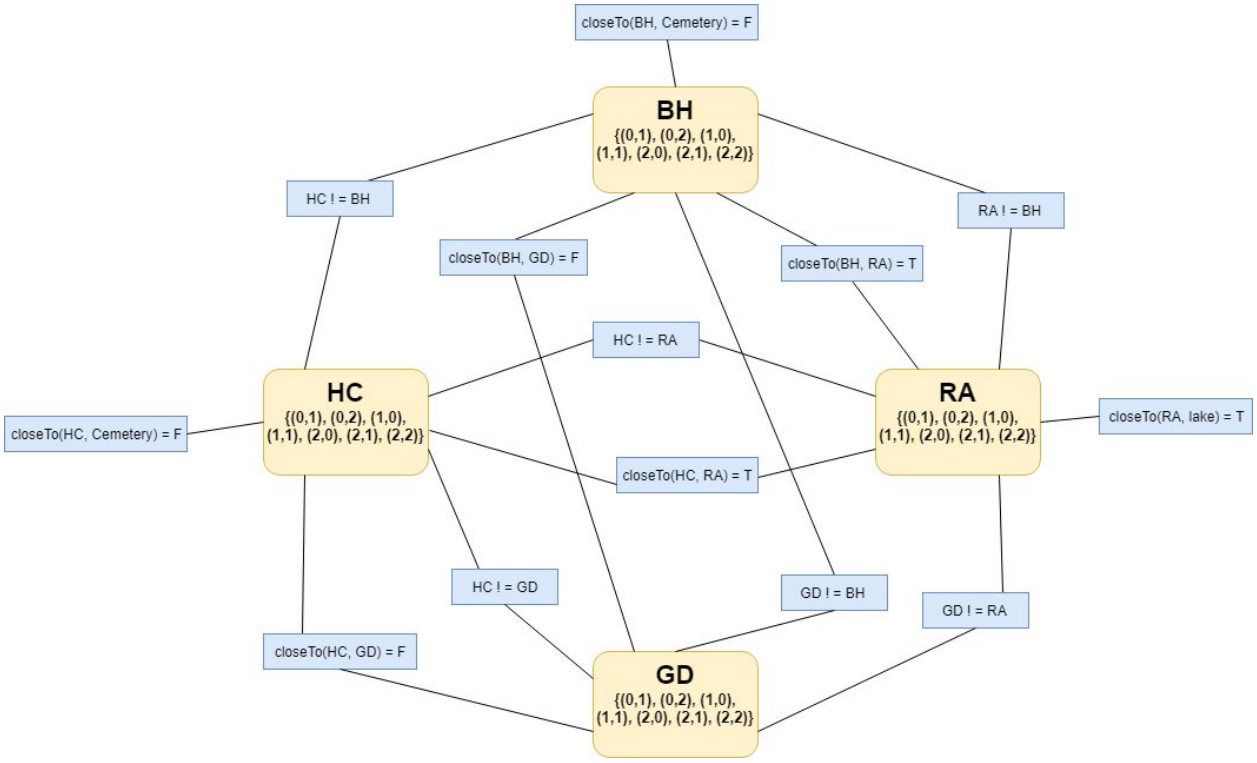
a)

- Variables: housing complex (HC), big hotel (BH), recreational area (RA), garbage dump (GD).
- Domains: The four variables all have the same domains (all of the cells in the 3x3 grid OR all of the cells in the 3x3 grid except 0,0 and 1,2)
- Possible worlds: All the ways of assigning all of the variables (development areas) to one cell of the 3x3 grid.
- Constraints:
    - First we define a function "closeTo" where if two development areas v1 and v2 are in cells that share an edge, then closeTo(v1, v2) = True:

```
function closeTo(v1, v2) {

      if (v1.column == v2.column) {
            if (v1.row == v2.row +1 || v1.row == v2.row -1) {
                  return true;
            }
      }

      if (v1.row == v2.row) {
            if (v1.column == v2.column +1 || v1.column == v2.column -1)
      {
                  return true;
            }
      }

      return false;
}
```

    - HC ≠ BH ≠ RA ≠ GD (the development areas cannot be in the same cell)
    - HC, BH, RA, GD ≠ 0,0 (only needed if this cell is included in the var domains)
    - HC, BH, RA, GD ≠ 1,2 (only needed if this cell is included in the var domains).
    - closeTo(HC, Cemetery) = F
    - closeTo(BH, Cemetery = F
      closeTo(RA, lake) = T
    - closeTo(HC, RA) = T
    - closeTo(BH, RA) = T
    - closeTo(HC, GD) = F
      closeTo(BH, GD) = F

b) not including 0,0 or 1,2 in the var domains:



Question 2:

a. Solutions:
A=1 B=3 C=2 D=3 E=1 F=4 G=1 H=2
A=2 B=2 C=3 D=4 E=2 F=1 G=2 H=3
A=3 B=1 C=4 D=3 E=1 F=2 G=3 H=4
A=3 B=3 C=4 D=3 E=1 F=2 G=3 H=4
total failures: 1278

b. Some examples:
Heuristic: Count the number of constraints that each variable is a part of. Expand them in the order from largest to smallest, break ties arbitrarily.

Variable	A	B	C	D	E	F	G	H
#constraints	2	1	5	5	4	6	5	6

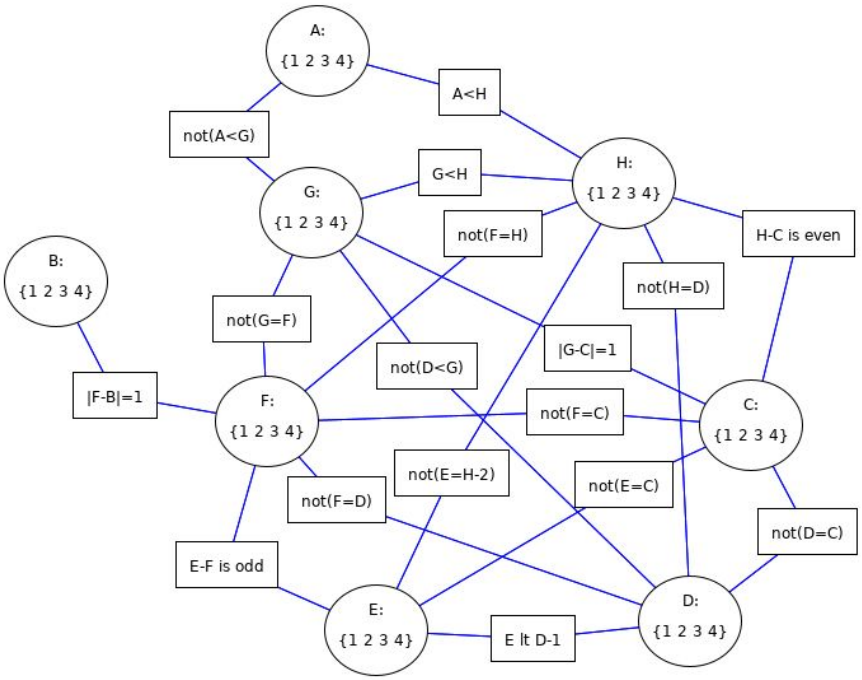
Example ordering: FHCDGEAB

Heuristic: Choose the most constrained variable, then choose the variables that are in the most constraints with previously assigned variables.

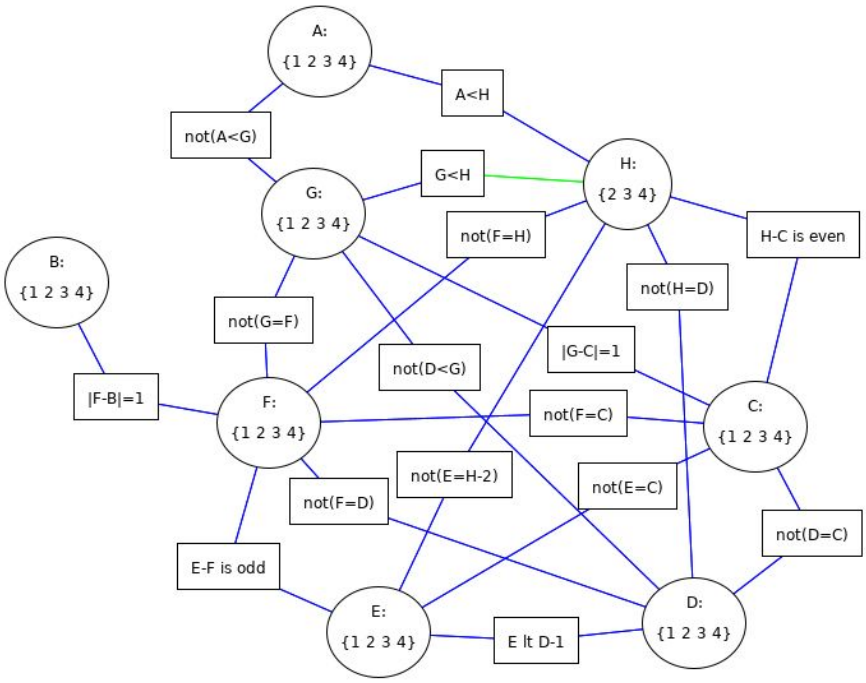
c. Pruning the tree earlier can expand fewer nodes. We cannot check a constraint until all variables in the constraint are assigned a value. If we expand a variable that has more constraints first, it is likely that we will find the inconsistencies sooner.

Question 3:

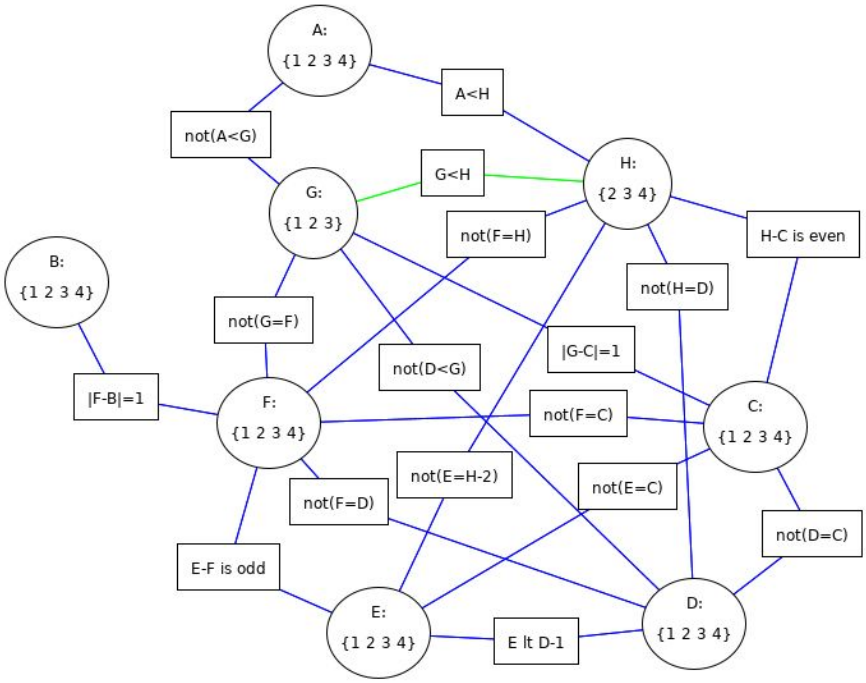
a)



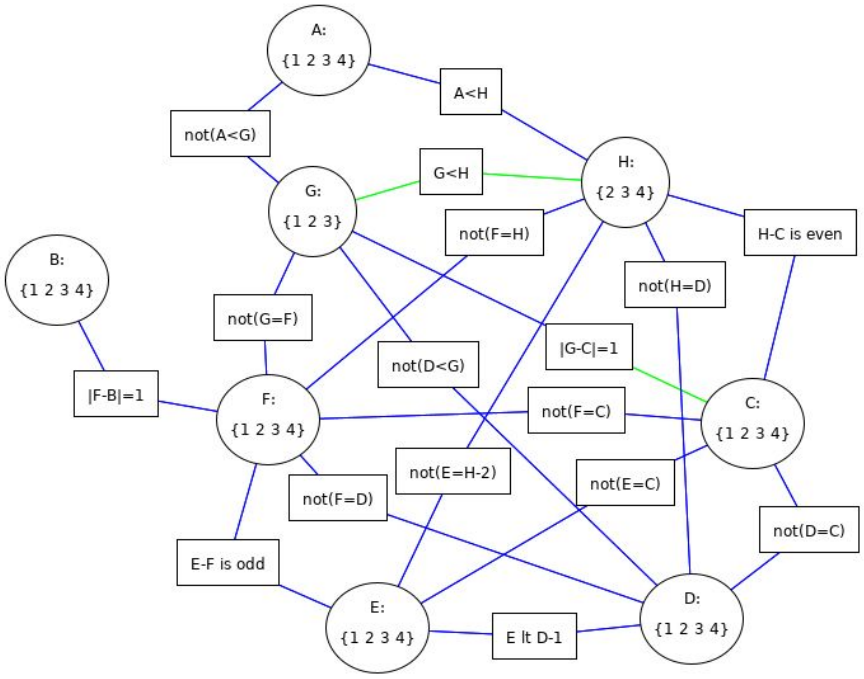
Initial constraint graph



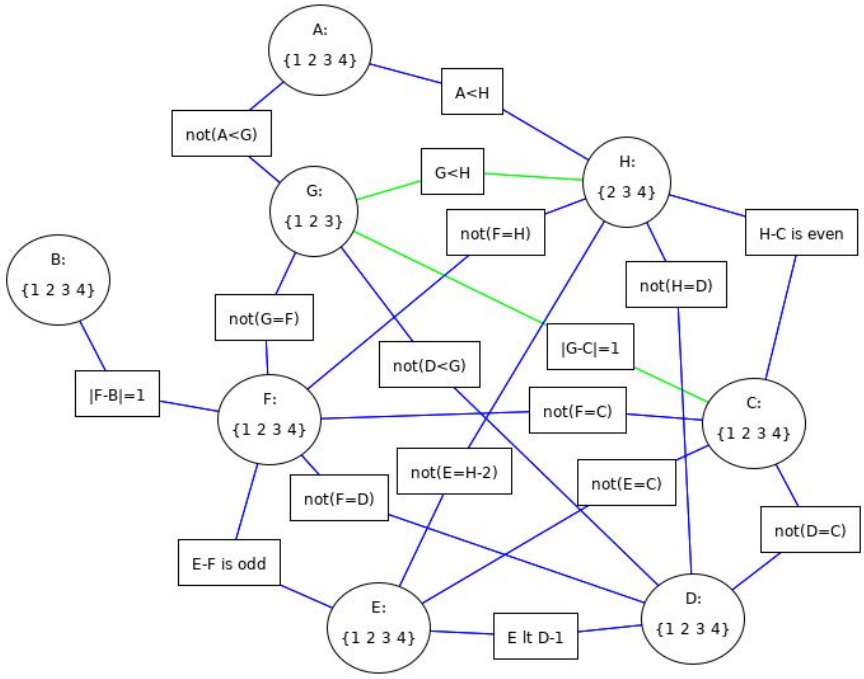
1) Element 1 has been removed from the domain of H.



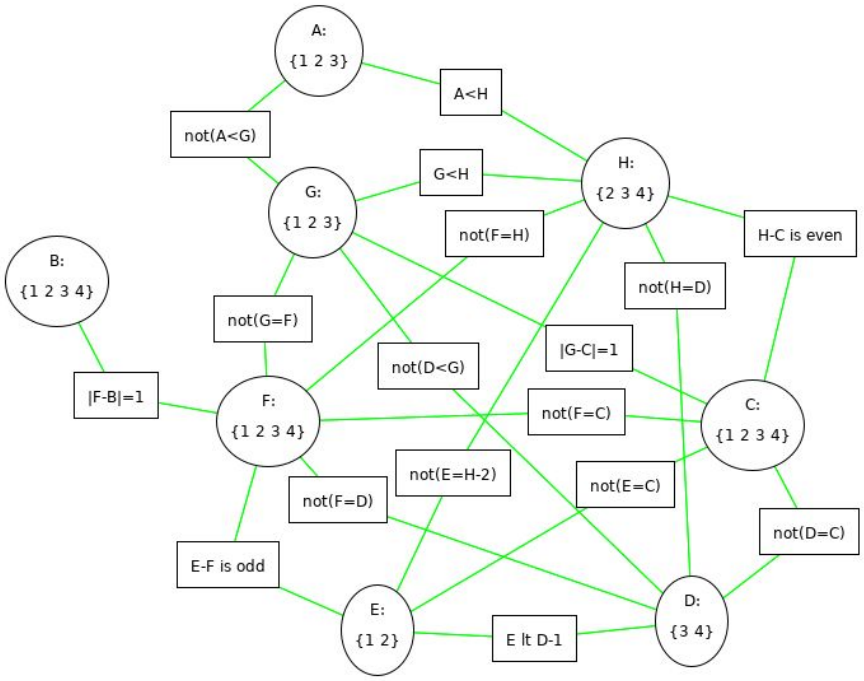
2) Element 4 has been removed from the domain of G.



3) No element has been removed from the domain of C.

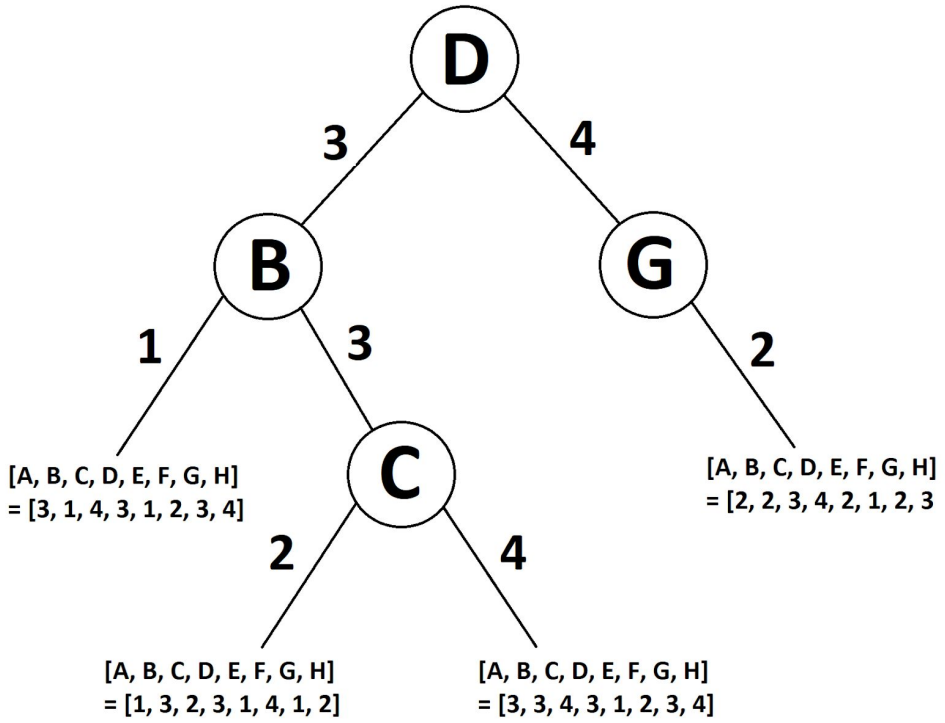


4) No element has been removed from the domain of G.



Constraint graph after arc consistency has finished.

b)
This is an example of different splitting trees:



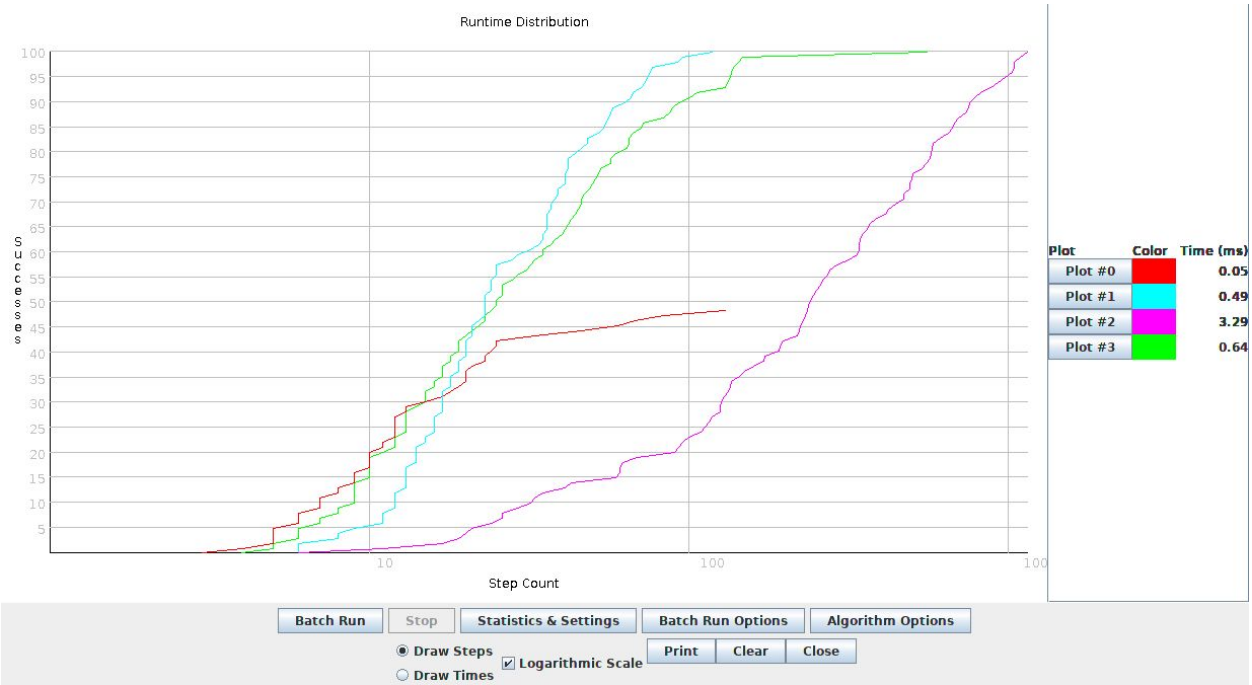
c)
With DFS the states are leaner (partial assignments), so the space complexity can be better, but (if there is not much pruning) the time complexity can be really bad. With the other option, time is relatively good (polynomial) for arc consistency, but the states in domain splitting are full CSP problems (more space).

Question 4:

a)
Show trace gives:
Initialization: [A, B, C, D, E, F, G, H] = [2, 4, 4, 1, 2, 4, 3, 1]

C --> 3, leaving 9 unsatisfied constraints
G --> 2, leaving 7 unsatisfied constraints
D --> 2, leaving 5 unsatisfied constraints
B --> 3, leaving 4 unsatisfied constraints
G --> 1, leaving 5 unsatisfied constraints

b)



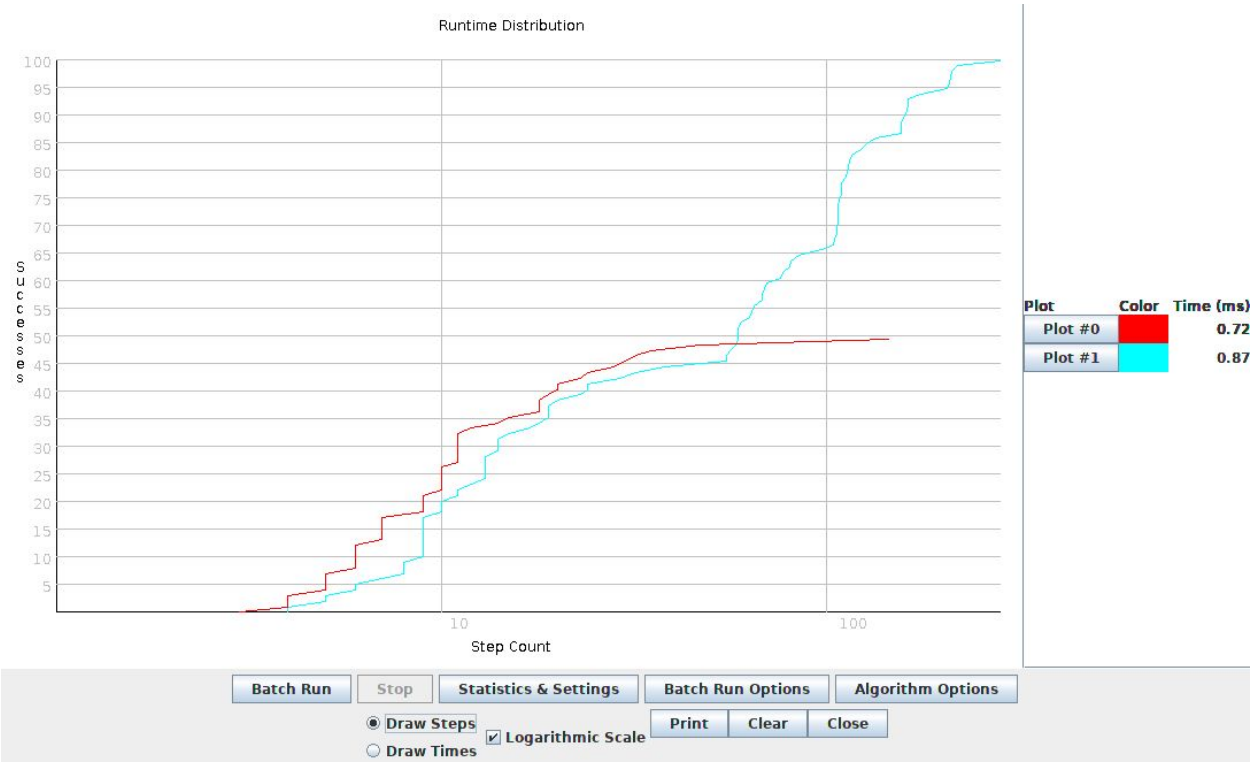
(i) is red, (ii) is light blue, (iii) is magenta, and (iv) is light green (using a 70/30 mix of i and ii). Note that the best variable/best value method never reaches 100 because it gets stuck in local minima. Choosing variables randomly performs quite poorly overall but doesn't seem to have a problem with local minima. Choosing the probabilistic mix performs very well, but it is slightly

outperformed by the method of selecting any variable involved in unsatisfied constraints. Note that early on, the best/best and probabilistic mix methods are comparable to each other and outperform the others; this suggests that selecting the “best” variable is usually the best choice, and that occasionally choosing other variables involved in unsatisfied constraints prevents long-term issues with local minima.

c)
Answers will vary; but one thing that should be noticed is that the plots from different batch runs do not overlap perfectly. This is because of the stochastic nature of the methods; since individual runs differ in how quickly they are solved, batches of runs will differ in how many of their runs will find a solution within x steps. This means that we can’t necessarily take a single batch run as producing the exact performance of the algorithm, but only as an approximation. To get better approximations, we could aggregate a number of batch runs, or (equivalently) we could do a single batch containing a larger number of runs.

d)
For each of these, choosing a random value works much worse than choosing the best value. The search is not directed by the scoring function

e)



Allowing random restarts greatly improves the performance of the best/best method in the long term, since it allows the search to leave local minima; however, comparing it to the previous plot, the performance still is not as good as methods (ii) and (iv).