CS171 Project Progress Final Report

My name: Trung Nguyen
ID# 14853167
UCInetID: trungtn@uci.edu

By turning in this assignment, I/We do affirm that we did not copy any code, text, or data except CS-171 course material provided by the textbook, class website, or Teaching Staff.

Part I

The programming language(s) and versions you used in your project: Python 3.3

The environment needed to compile and run your project: Openlab machine / Ubuntu Linux with python3

A small write up of your implementation:

Heuristic Name	Command Line Abbreviation	Heuristic Type	Description
Forward Checking	FC	Consistency Check	On an assignment of a value to a variable, each of the neighbors of the variable in the constraint graph will have inconsistent values removed from their domain.
Minimum Remaining Values	MRV	Variable Ordering	The next variable to be assigned a value will be the variable with the

			fewest number of legal moves remaining.
Degree Heuristic	DH	Variable Ordering	The next variable to be assigned a value will be the variable which is involved in the most number of constraints with other unassigned variables.
Least Constraining Value	LCV	Value Ordering	The next value to assign to a variable will be selected based on the number of constraints it places on other unassigned variables in the constraint graph, with the value causing the least number of constraints selected.
Arc Consistency	AC	Consistency Check	Arc consistency eliminates values from domain of variable that can never be part of a consistent solution.

I/We coded it.			I/We tested it thoroughly.			It ran reliably and correctly.			What was it?
Yes	Partly	No	Yes	Partly	No	Yes	Partly	No	
Required Coding Project									
X			X			X			Backtracking Search (BT)
X			X			X			Forward Checking (FC)
X			X			X			Minimum Remaining Values (MRV)
X			X			X			Degree Heuristic (DH)
X			X			X			Least Constraining Value (LCV)
Extra Credit									
X			X			X			Writing Your Own Shell
									Writing Your Own Random Problem Generator
X			X			X			Arc Consistency AC-3/ACP/MAC
									Local Search using Min-Conflicts Heuristic
X			X			X			Advanced Techniques, Extra Effort, or Creativity Not Reflected Above (*)

(*) Advanced Techniques, Extra Effort, or Creativity Not Reflected Above:

I wrote the small script automation in python3 to test all the test cases thoroughly.
Run 'python3 runtest.py' located in folder testfiles.

Part II

FC	MRV	DH	LCV	AC	AVERAGE # NODES	AVERAGE TIME	STD DEV. TIME
					390K+	1800+ timeout	0
X					5866.33	746.56	931.90
	X				159411	600+	0
		X timeout			150K+	3600+	0
			X		215614	1800+	0
				X timeout	1308	900+	0
X	X	X	X		3117.66	312.11	451.57
X	X				5782	607.32	1032.89

(*) Note to TA:
For part II: I got '***timeout***' on ***PH3 and PH4*** so I excluded them in these hard test cases. Only ***PH1 + PH2 + PH5 works*** for me under python3.

You can find all my result outputs under folder ***outputfiles***; I keep them as proof.

Did you get the results you expected?
Why or why not?

Answer:
Yes, 70-80% reasonable
I feel like if I have more time, I could finish running them. I believe my results are ***fast*** even with under ***limitation*** of Python ***slowness!***

Did you implement any Advanced Techniques, Extra Effort, or Creativity not reflected
above? If so, please tell us what you did.

Answer:
I wrote the small script automation in python3 to test all

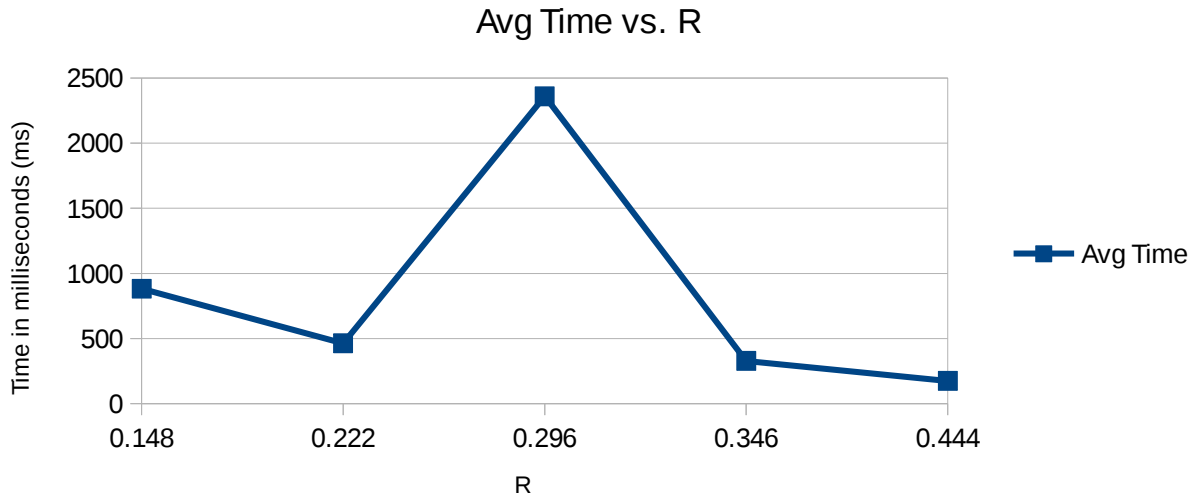
the test cases thoroughly.
Run 'python3 runtest.py' located in folder ***testfiles.***

Part III

Base on my result, I'm going to run FC-MRV-DH-LCV:

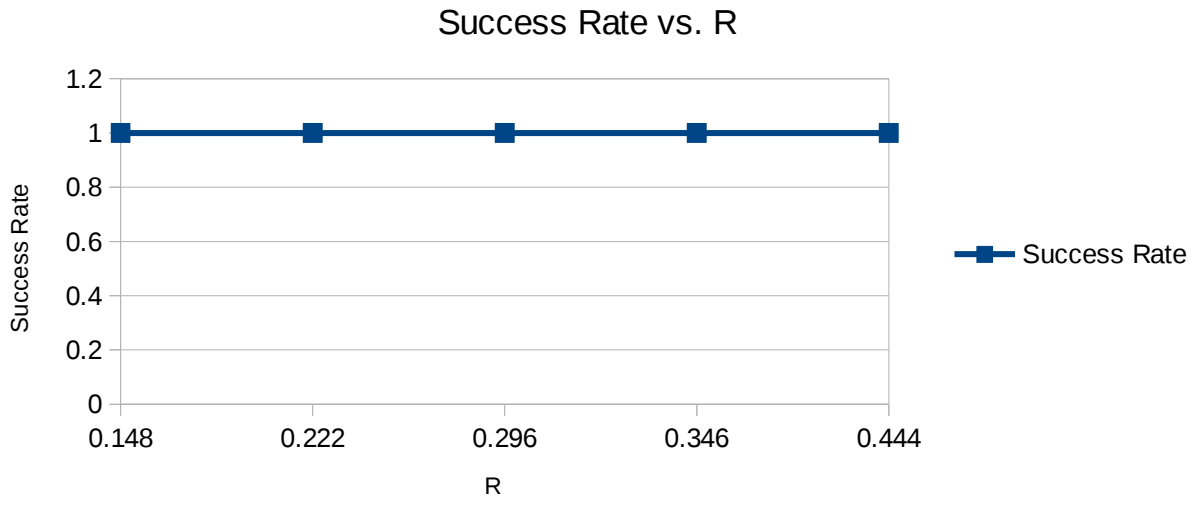
M	N	P	Q	R=M/N^2	AVERAGE #NODES	AVERAGE TIME	STD. DEV. TIME	# (%) SOLVABLE
4	9	3	3	0.0494				
8	9	3	3	0.0988				
12	9	3	3	0.148	59.66	0.8825	0.34	100
16	9	3	3	0.198				
17	9	3	3	0.210				
18	9	3	3	0.222	29.33	0.463	0.1122	100
19	9	3	3	0.235				
20	9	3	3	0.247				
21	9	3	3	0.259				
22	9	3	3	0.272				
24	9	3	3	0.296	190.33	2.36	1.8	100
28	9	3	3	0.346	25.66	0.3279	0.1281	100
32	9	3	3	0.395				
36	9	3	3	0.444	14	0.1748	0.1223	100

3.2. Find the critical value of the "hardest R" for N = 9 and your best combination above.



Based upon your results above, you estimate the "hardest R_9" = 0.29

3.3. How does puzzle solvability for your best combination vary with R = M / N^2?



3.3 Is the critical value for "hardest R" approximately the same as the value of R for which a random puzzle is solvable with probability 0.5?

I dont understand this question ?!
My solver is probability for 100% = 1.0

Part IV

Base on my result, I'm going to run FC-MRV-DH-LCV:

"Hardest M" round (N^2 x R_9)	N	P	Q	#(%) Completed in 5 Minutes or Less	AVERAGE #NODES	AVERAGE TIME	STD. DEV. TIME
144*0.29 = 41.76	12	3	4	60%	3699.8	138.33	149
	15	3	5				
256*0.29 = 74.24	16	4	4	40%	9914	200.87	137.94
	18	3	6				

	20	**4**	**5**				
	21	**3**	**7**				
	24	**4**	**6**				
	27	**3**	**9**				
	28	**4**	**7**				
	30	**5**	**6**				
	32	**4**	**8**				
	35	**5**	**7**				