CSC520 - Artificial Intelligence

Assignment 2

Name: Sai Vishnu Muvvala

Unity ID: smuvval

Question 1:

Given specification of sentences:

1. q -> g
2. q -> n
3. q ^ r -> i
4. q -> oc v ic
5. oc ^ ad -> i
6. p ^ q
7. ¬i -> p ^ oc ^ ¬ad ^ ¬r (peter(p)+lives oncampus(oc)+no entry in dorms(¬ad)+no entry in library(¬r))
8. p ^ ¬n

Question a: The specification is not consistent. Here's why:

9. q (And-Elimination from 6)
10. n (Modus-Ponens from 2,9)
11. ¬n (And-Elimination from 8)
12. n ^ ¬n (And-Introduction from 10,11)

In the above, point 12 (n ^ ¬n) is a contradiction as it is false in every case. Hence, given specification is not consistent.

Question b: If Peter is able to login to MyPack website, Point 8 above changes as: p ^ n, eliminating inconsistency. It is consistent. Below is truth-table for a model.

p	q	n	g	r	i	oc	ic	ad	1	2	3	4	5	6	7	8
T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T

In the above truth-table, 1,2,3,4,5,6,7 refer to sentences stated above. 8 refers to the modified sentence, which is, Peter can log in the My Pack website. So, 8 refers to p ^ n.

Question 2:

Note: For BackTracking, I have implemented a special heuristic called **'Minimum Remaining Values(MRV)'**. The job of this heuristic is to pick the variables with smaller domains first. By using this, I was able to eliminate huge amount of state space search and unwanted backtracks.

1. **Class Assignment Problem:**

BackTracking: Without using MRV heuristic stated above, the solution was taking 2 backtracks. By using MRV, the solution was taking zero number of backtracks.

MinConflicts: The number of steps to obtain solution varied widely from 1 to 63. Even though the approach consumes lesser memory, 63 steps for obtaining solution is quite high. Having more number of constraints might have helped.

2. Map Coloring Problem:
BackTracking:
Smaller graph(Map_v) Without MRV heuristic: 0
Smaller graph(Map_v) With MRV heuristic: 0 backtracks
Larger graph(Map_u) Without MRV heuristic: 47
Larger graph(Map_u) With MRV heuristic: 0 backtracks
MinConflicts: In the numerous runs that I made, the larger graph(Map_u) has found a right assignment in given max_steps=20. The steps for solution varied widely, ranging from 53 to greater than 300. The steps for smaller map varied from 5 to >20.

As there are many possible right combinations in this problem, BackTracking can give right assignment in lower number of steps.

3. Sudoku Problem:
Without using MRV heuristic: It took 1121 backtrack steps to find valid combination.
With using MRV heuristic: It took **Zero** backtrack steps to find correct solution.

MinConflicts: In my test runs, even with 3000 max_steps permit, MinConflicts was able to solve 68-72 variables only. From this we can definitely infer that, larger the number of variables, lesser is the probability of finding solutions using MinConflicts. As there are too many number of variables that play randomly, MinConflicts shall most probably take large number of steps to obtain solution. **All we can talk about is the probability of attaining solution only. We can never declare that MinConflicts cannot develop a cheap solution. In its luckiest situation, it might give a valid solution in O(1) time.**

Even though there are as many as 81 variables in this problem, because of the numerous conditions and the given assignments, backtrack, with a heuristic was able to find solution for this problem using zero backtracks.

4. Zebra Problem:
Without using MRV heuristic: 21992 backtracks
With using MRV heuristic: 466 backtracks
Here's perfect example for brute force of backtrack. Without using MRV, backtrack took as many as 21k backtracks. With heuristic, it took only 2% of the backtracks(466).
MinConflicts: Again, MinConflicts was able to sometimes find solution in as minimum as 262 steps. Other times, 3000 steps weren't sufficient to find a solution.

Comparison between BackTracking and MinConflicts:

Both the algorithms are good and bad in their own ways. Some of my key observations are:

1. BackTracking definitely finds a solution, if there is any. It might consume heavy resources when proper heuristics were not used, but it will eventually find the solution.
2. Without heuristics, **BackTracking is a Brute force approach**. As stated above, for Zebra problem, there were as many as 22k backtracks without using heuristic. By using **Minimum Remaining Values heuristic**, the backtrack count was cut down to 466.
3. A general BackTracking approach can create a huge toll on memory resources. Imagine a situation where there are 10k variables. If the first assignment of a variable is wrong, and we realized it only in the end, the recursion heap memory should support all the 10k backtrack calls. It can create a huge mess.
4. If there are many right answers, the backtracks are less in number. If there is only one possible assignment, like in Zebra/Sudoku, the backtracks are many.
5. MinConflicts doesn't demand as much memory requirement as BackTracking does.
6. MinConflicts is good at finding local optimal solutions, which might not be right most number of times.
7. The initial assignments to variables is random. Also, tie-breaking between two potential solutions(when both have same nummber of min conflicts) is random. Hence, there's heavy amount of randomness involved in this.
8. As randomness doesn't work with equal probabilities, in the worst case situation, the same wrong assignments might reoccur numerous time. MinConflicts might never move ahead. There's possibility for deadlocks.
9. Even with all these negative attributes of MinConflicts, it might turn out to be best in some cases. If there are problems with a little number of variables, a larger number of constraints and smaller domains, MinConflicts approach can prove to be way better than the Brute Force style BackTracking approach.

How could these approaches be developed?

1. BackTracking: MRV heuristic proved to be of great worth, as it has eliminated huge number of backtracks. If the BackTrack was further improved, using the features of MinConflicts such as '**Least Constraining Value** - Picking value in the domain with least conflicts', it could turn out to be a hybrid algorithm, giving the best of both the worlds.

2. MinConflicts: MinConflicts can be improved by modifying the way we make the initial random assignments. If we developed a heuristic to **first assign values to variables which have a larger number of constraints**, it would perform better. Imagine the other way. In a problem there are variables with huge number of constraints, least number of constraints. For the higher degree variables, the number of correct options in its domain might predictably be smaller(because of higher degree). If those right options were robbed off by variables of lower degree, the available right options for higher degree variables shall further shrink, leading to wrong assignments and delays in obtaining solutions.