CSE: 537 Artificial Intelligence

Homework 2

Name: Leena Shekhar Sbu Id: 110944612

Question 1: Performance analysis of the following algorithms to solve Map coloring CSP problem:

1. DFSB- Depth First Search with Backtracking
2. DFSB++- Depth First Search with Backtracking modified to incorporate Variable and Value ordering and ARc Consistency.
3. Minimum Conflicts- Local search algorithm with minimum number of conflicts heuristic used.

Test Cases	**DFSB**	**DFSB++- MRV, LRV, Arc Consistency**	**Min Conflicts**
backtrack_1	No answer	Search calls: 153 Arc pruning calls: 197 Time taken: 0.10914134979248047	No answer
backtrack_2	No answer	Search calls: 201 Arc pruning calls: 536 Time taken: 0.15468311309814453	No answer
backtrack_easy	Search calls: 9 TIme taken: 7.796287536621094e-05	Search calls: 9 Arc pruning calls: 13 Time taken: 0.0005404949188232422	Steps taken: 225 Time taken: 0.0066487789154052734
backtrack_hard	No answer	Search calls: 42139 Arc pruning calls: 787885 Time taken: 17.24136185646057	No answer
minconflict_1	Calls needed: 22 Time taken: 0.00019311904907226562	Search calls: 13 Arc pruning calls: 21 Time taken: 0.0020866394042968875	Steps taken: 3643 Time taken: 0.1464381217956543

minconflict_2	Calls needed: 13 Time taken: 0.00023889541625976562	Search calls: 13 Arc pruning calls: 20 Time taken: 0.0011379718780517578	Steps taken: 9230 Time taken: 0.33767104148864746

*Algorithms were run for 1 minute to get the output. If no solution was found till then then “No answer” was produced as the output.

Question 2: Observations made while coding the algorithms

1. DFSB: THis algorithm chooses one variable at a time and assign values to that variable. This algorithm is better than naively assigning values to all variables at every levels as rather than n! D^n it expands D^n leaves. By just considering commutativity we can reduce the search space by a huge factorial factor. It used the current assignment to generate the successors. But since we know that plain backtracking is uninformed, it does not perform well for large problems. We can see the performance of DFSB in the table above to cross check this hypothesis.

2. DFSB++: We know that we can improve the performance of uninformed search methods by incorporating some knowledge about our understanding of the problem domain. DFSB++ is in improvement of plain DFSB which the following knowledge has been added:
   2.1. Variable ordering: This decides which variable is to be selected next for assignment. The most restricted variable is chosen so that if the problem cannot be solved, we reach failure state faster. THus choosing the variable with the fewest “legal” values- is called Minimum remaining value or MRV.
   2.2. Value ordering: This decides in what order is the value to be assigned to the variable thus chosen so that the assignment leads to pruning of minimum number of values for the neighboring variables.
   2.3. Arc consistency: This is a way of propagating constraints and is better in performance than forward checking as it removes all the inconsistencies in the graph. The worst case time for the algorithm is $O(n^2d^3)$, which is a little costlier than forward checking but nonetheless better in performance.

   We can see from the table above that performance of DFSB is really good compared to DFSB plain. DFSB++ performs really well on problems of decent size and with many constraints as the algorithm backtracks fast and leads to a goal state.

3. Minimum conflicts: This algorithm initially assigns a value to every variable and then successively changes the value of one variable at a time such that that value results in minimum number of conflicts with other variables.
   For Min conflict convergent depends a lot on initial assignment of values and randomness incorporated to make sure that the agent does not get stuck in the local minima. The algorithm is run for a maximum number of steps and if the solution has not been reached after a particular number of steps then randomly a few variables are chosen and their values are changed to avoid local depressions. Also, conflicted variables are chosen randomly to boost the performance of the algorithm by avoiding repetition as it was observed that same variables were getting repeated many times in the conflicted variables list.
   From the table above we can observe that Min conflict algorithm takes more time than that of DFSB++, but performace at this point cannot be exactly compared as MIn conflict is a random algorithm.

   Min conflict performs really well when the problem domain is huge in size but less constrained while DFSB++ performs really well when the problem domain is highly constrained but is of decent size.

References:

1. Lecture slides and class notes
2. Artificial Intelligence book by Peter and Norvig
3. Wikipedia for pseudocode.
4. Stackoverflow for Python related code