Assignment 2 Report

Name: Swaminathan Sivaraman
SBU ID: 110951180
Course: Artificial Intelligence CSE 537
University: Stony Brook University

Section 1.
A table describing the performance of the algorithms (DFSB, DFSB++, and MinConflicts) on the sample problems.

The below table compiles the performance of the DFSB, DFSB++ and MinConflicts algorithms for the sample problems given. The problem is marked 'Timed out' if an algorithm can't solve it under 60 seconds.

For DFSB, steps taken = number of search calls
For DFSB++, steps taken = number of search calls + number of arc pruning calls
For MinConflicts, steps taken = number of search steps (states explored)

Input File	Algorithm	Time Taken (ms)	Steps taken
backtrack_easy	DFSB	0.042	8
	DFSB++	1.241	16
	MinConflicts	0.117	6
backtrack_hard	DFSB	Timed Out	5694995
	DFSB++	11358.897	1002
	MinConflicts	4584.130	5064
backtrack_1	DFSB	Timed Out	7908722
	DFSB++	449.613	240
	MinConflicts	534.891	2529
backtrack_2	DFSB	Timed Out	7880854
	DFSB++	1328.221	401
	MinConflicts	1674.098	5031
minconflict_easy	DFSB	5.908	1505
	DFSB++	11.086	50
	MinConflicts	2.379	74
minconflict_hard	DFSB	Timed Out	8640902
	DFSB++	441.232	240
	MinConflicts	208.317	977
minconflict_1	DFSB	0.128	28
	DFSB++	2.488	24
	MinConflicts	0.658	32
minconflict_2	DFSB	0.063	12
	DFSB++	2.583	24
	MinConflicts	1.719	89

Section 2.
Explain the observed performance differences

DFSB:
As seen, the naïve DFSB algorithm cannot solve most of the problems in reasonable time. Even when it can solve the problem, it explores a lot of states when compared to the other two algorithms (eg: 1505 states explored by naïve DFSB for minconflict_easy while only 50 and 74 states explored by DFSB++ and MinConflicts).

DFSB++:
This greatly improves upon the performance of the naïve DFSB algorithm. It can solve all the problems that the original algorithm couldn't. There is a small overhead in doing the arc-pruning call at each step but as the algorithm reduces the overall number of steps exponentially, this does not affect the running time too much. The reduction in the number of steps can be clearly observed in the same minconflict_easy case. While naïve DFSB took 1505 steps to find the solution, DFSB++ could find the solution in 50 steps itself.

MinConflicts:
The implemented Minconflict algorithm uses random restarts to avoid issues like local maxima, ridges and plateaus. This algorithm is able to find the solution for all cases even if run repeatedly, though the running times differ. Although, for the problems given, the running times did not vary too much across multiple runs. It can be seen that the MinConflicts algorithm explores slightly more number of steps than DFSB++. However, we know that the running time of DFSB++ will definitely grow larger as the number of variables increases. MinConflicts however can find the solution possibly faster, if we can show that solutions are distributed more or less uniformly across the state space.