ANALYSIS

Description of the strategy:

Strategy 0: This strategy has no heuristic and choosing the next move randomly from the valid ones. It searches deeply into the random move at each step. The chance to get the right solution from these sequence of random picks is very small. Moreover, considering the high dimensionality, the machine buried itself deeply into the “wrong” and can rarely return a right solution.

Strategy 1: This strategy uses the simple heuristic based on the notion of fixed degree of the squares, which describes the number of neighbors on the board of each square. This strategy always pick randomly from the neighbors with the lowest fixed degree firstly to search for possible solutions.

Strategy 2: This strategy uses the simple heuristic based on the notion of dynamics degree of the squares, which describes the number of non-visited neighbors on the board of each square. This strategy always pick randomly from the neighbors with the lowest dynamics degree firstly to search for possible solutions.

Strategy 3: This strategy uses the simple heuristic based on the notion of fixed degree of the squares, which describes the number of neighbors on the board of each square. Different from stratehy 1, before it moves to the next step, it will firstly check the dynamics degrees of the unvisited neighbors of the current square. If there are more than one such neighbor with the dynamics degree of 1, it will directly recognize it as a failure and return to its previous backup. Otherwise, this strategy always pick randomly from the neighbors with the lowest fixed degree firstly to search for possible solutions.

Strategy 4: This strategy uses the simple heuristic based on the notion of fixed degree of the squares, which describes the number of neighbors on the board of each square. Different from stratehy 1 and 3, before it moves to the next step, it will firstly check the dynamics degrees of the unvisited neighbors of the current square. If there is exactly one such neighbor with the dynamics degree of 1, it will directly move into this square it and will not try any other neighbors after it returns from searching this neighbor (with dynamics degree of 1). Otherwise, this strategy always pick randomly from the neighbors with the lowest fixed degree firstly to search for possible solutions.

Strategy 5: This strategy is a combined version of 2,3,4.
This strategy uses the simple heuristic based on the notion of dynamics degree of the squares, which describes the number of neighbors on the board of each square. Different from stratehy 1 and 3, before it moves to the next step, it will firstly check the dynamics degrees of the unvisited neighbors of the current square. If there are more than one such neighbor with the dynamics degree of 1, it will directly recognize it as a failure and return to its previous backup. If there is exactly one such neighbor with the dynamics degree of 1, it will directly move into this square it and will not try any other neighbors after it returns from searching this neighbor (with dynamics degree of 1). Otherwise, this strategy always pick randomly from the neighbors with the lowest dynamics degree firstly to search for possible solutions.

Strategy 6: (My strategy)
This strategy is nearly the same as strategy 5. Except for the first 2 steps.
The start squre is at (2,3). I firstly set the first two moves as:
(2,3) -> (1,1) -> (3,2).

Then from (3,2), I used exactly the same approach of strategy 5. Moreover, each time when a solution found, I generated another solution which is the inverse path of this found solution.
Since I have specified the direction of the first two steps, any generated solution will end with: (3,2) -> (1,1) -> (2,3). Thus any generated solution should be different from any of the program found ones. Moreover, since the program found ones are different with each other, the generated solutions should be different from each other as well. Therefore, all the solutions are distinct.

Summary data

Chart 1: Number of moves per second for strategy 1-6. (S6 is my strategy)
Blue bars denote the mean values and red bars denote the median value. Error bars in black are the standard deviation.

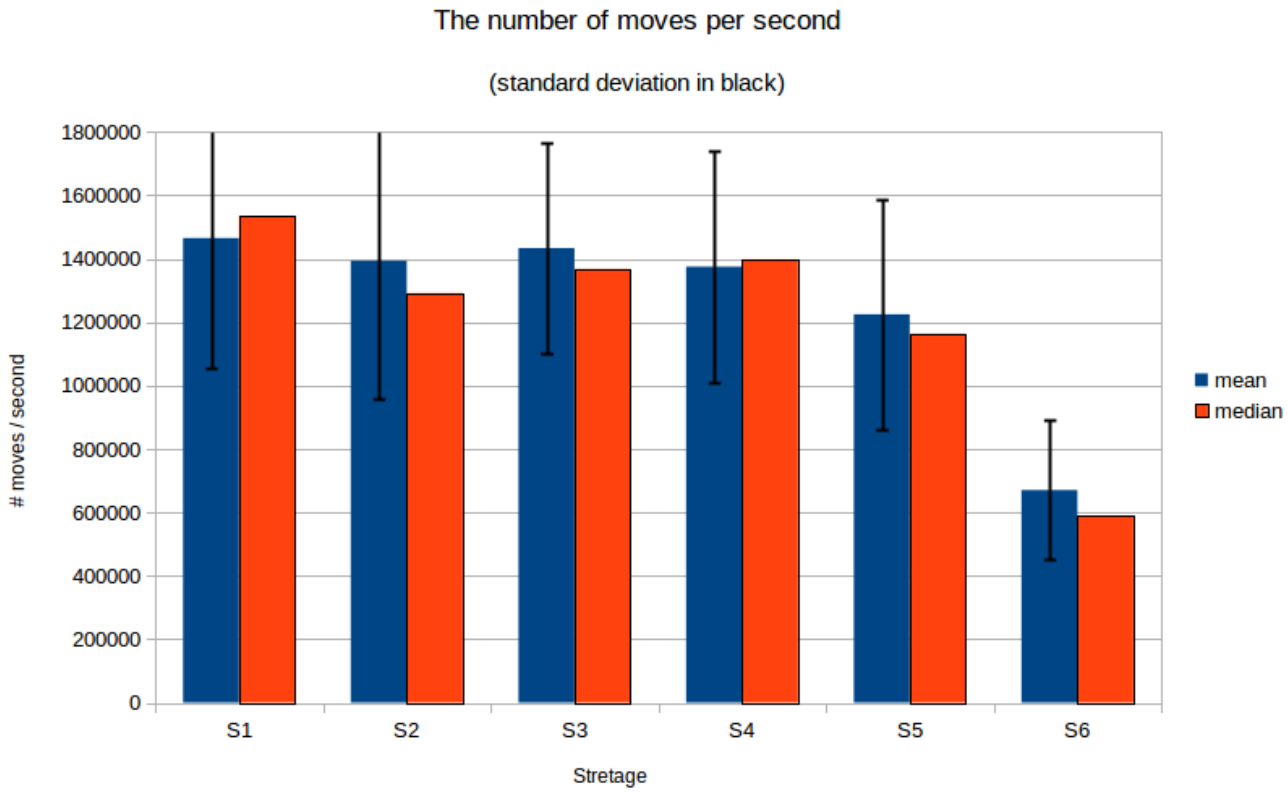

Chart 2: Number of sols per second for strategy 1-6. (S6 is my strategy)
Blue bars denote the mean values and red bars denote the median value. Error bars in black are the standard deviation.

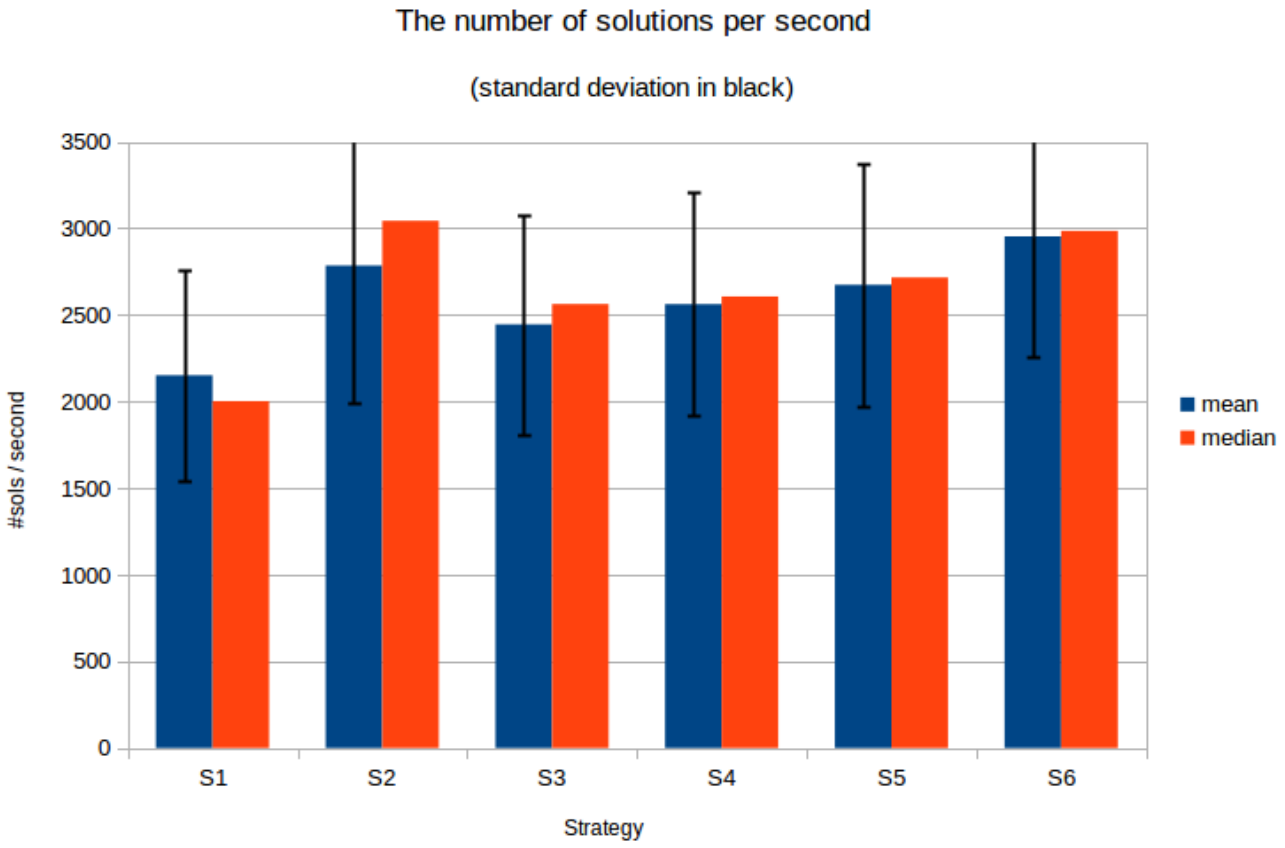

Table: Statistics result of 30 runs for each strategy:

Num_moves/s		S1	S2	S3	S4	S5	S6
	Mean	1465194.2952	1393705.3408	1433393.8434	1375021.1119	1224758.9447	670362.93062
	Media	1535404.1722	1291485.5688	1367658.154	1398318.1551	1163363.6761	589764.80673
	Std.Dev	410953.34194	433789.61483	331869.08806	365990.6023	364133.53275	218775.97443
Num_sols/sec		S1	S2	S3	S4	S5	S6
	Mean	2151.733744	2784.8697127	2445.8694049	2561.7512978	2674.3956398	2.95E+03
	Media	2002.6493473	3043.7280799	2562.6527865	2606.7781029	2716.7047719	2984.6776151
	Std.Dev	607.22760815	791.46085827	633.27801911	643.43442689	703.26388955	698.66752705
Time elapse(s		S1	S2	S3	S4	S5	S6
	Mean	0.7470012904	0.7821249359	0.7332716015	0.7878138403	0.8843933027	1.6268337026
	Media	0.6518226245	0.774568822	0.731181589	0.7151564085	0.859599609	1.696093633
	Std.Dev	0.2450135103	0.2253356835	0.1650700015	0.2434199564	0.2504942087	0.447745429
num of sols		S1	S2	S3	S4	S5	S6
	Mean	1651.4666667	2784.8697127	1886.3333333	2111.9333333	2493.5666667	5071.0666667
	Media	1368.5	3043.7280799	1858.5	1904.5	2430.5	5269
	Std.Dev	883.94783484	791.46085827	864.45184279	1060.6429388	1212.7893095	2245.5972853

Analysis/takeaways from the graph:

Chart one shows that implementing more complicate heuristic will take more computation time for each move. However, by implementing heuristic to the searching strategy, it is more likely to get a

feasible solution and thus the time used to find each solution decreases. For example, strategy 1 uses the fixed degree as its heuristic and it doesn’t need to updates the degree for every move. Thus strategy 1 gets the most moves per second. However, this simple heuristic might be less likely than other strategies to find a feasible solutions and thus it has the least sols per second.