ECEN 5018 Spring 2014 Project Proposal **Vince Coghlan**

The basic 9x9 sudoku puzzle can be seen to be a potential game. If each empty square is a player in a game, with a utility determined by the amount of repetitions in a row, column, and square, we can see that if a player makes a move to increase the amount of repetitions in a row, then that other player in that row will also incur the same costs of repetitions. This idea of effecting everyone else the exact same as they effect you gives rise to a potential game. We can find the potential function as half of the amount of repetitions of everyone on the board. This is because if you were to raise your repetitions by 1, you are increaseing someone else's repititions by one. That is a total of 2 increases in global utility, and to maintain the equals sign that defines a potential game, we need it to be a 1 to 1 ratio. For a group of players N and an action $a \in \mathcal{A}$, we can find that the potential function is:

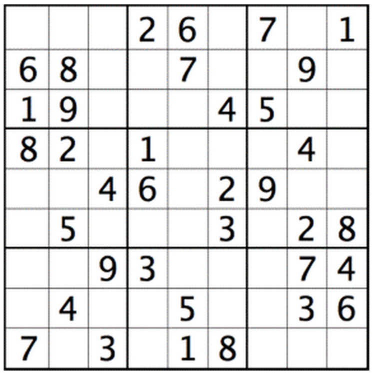
$$\Phi = \frac{1}{2} \sum_{i \in N} U_i(a)$$

This means that, for a temperature $T > 0$, implementing log-linear learning will induce a markov chain with a unique stationary distrobution of the form:

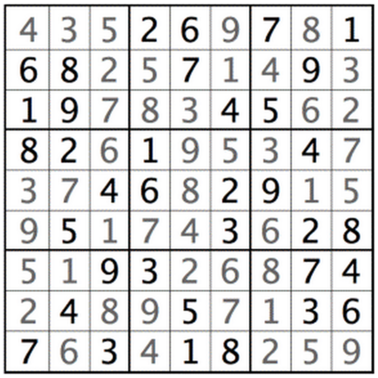
$$\pi_a = \frac{e^{\frac{1}{T}\Phi(a)}}{\sum_{\tilde{a} \in \mathcal{A}} e^{\frac{1}{2}\Phi(\tilde{a})}}$$

Knowing that a unique stationary distrobution can be found leads to some important questions. What about games with multiple solutions? How quickly will we arrive at this solution? What temperature will bring us there the fastest? What will happen in puzzles without solutions? Is this solution the stochastically stable state? Can we implement a more efficient solver by time-varying the temperature? Are there other learning algorithms that will give similar results?

I have implemented this log-linear learning solver in MATLAB, and it can solve most easy puzzles in a fairly short amount of time. For example, the following puzzle and soltion is usually solved within 50-100 iterations using a temperature of $\frac{1}{2}$:

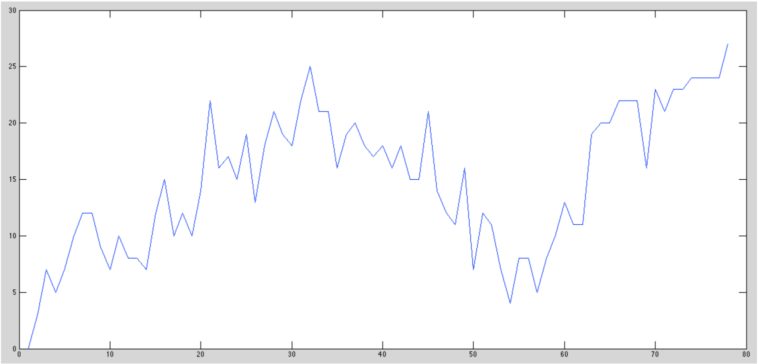


			2	6		7		1
6	8			7			9	
1	9				4	5		
8	2		1				4	
		4	6		2	9		
	5				3		2	8
		9	3				7	4
	4			5			3	6
7		3		1	8			

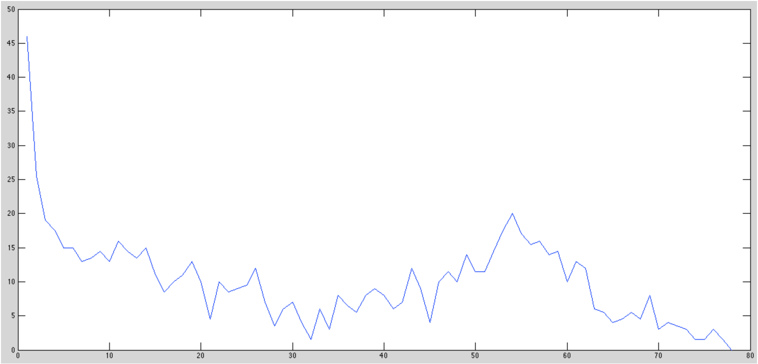


4	3	5	2	6	9	7	8	1
6	8	2	5	7	1	4	9	3
1	9	7	8	3	4	5	6	2
8	2	6	1	9	5	3	4	7
3	7	4	6	8	2	9	1	5
9	5	1	7	4	3	6	2	8
5	1	9	3	2	6	8	7	4
2	4	8	9	5	7	1	3	6
7	6	3	4	1	8	2	5	9

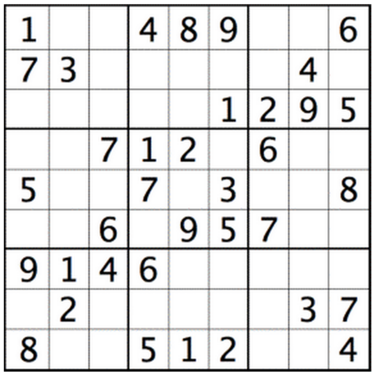
This is a graph of the iteration versus the amount of rows, columns, and squares that are complete.



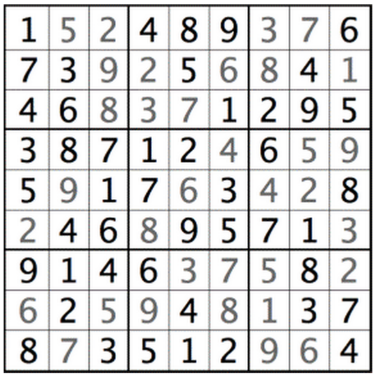
At 27, we are solved, since 27 is every row, column, and square complete. The potential can be seen below:



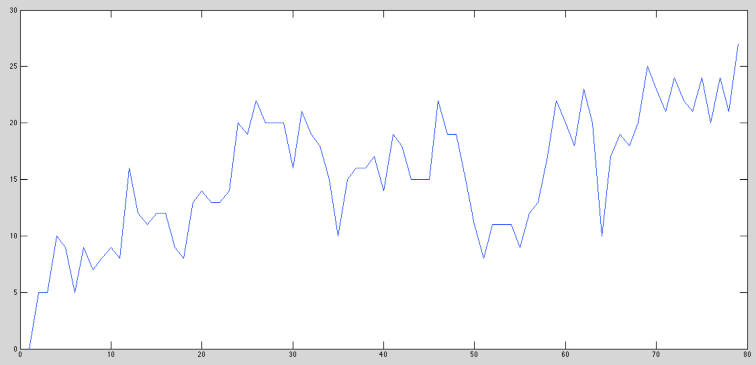
Here is another example that took 233 iterations to solve:

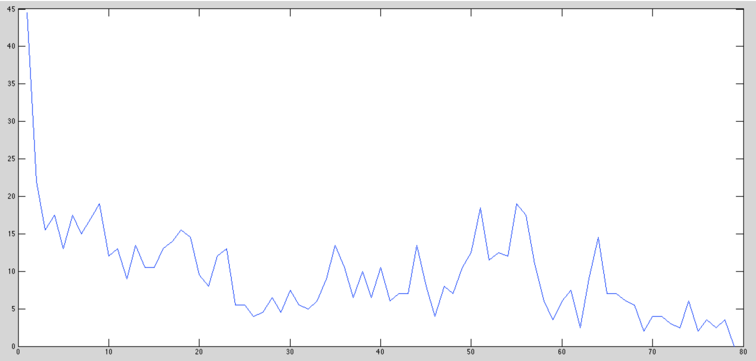


1			4	8	9			6
7	3						4	
					1	2	9	5
		7	1	2		6		
5			7		3			8
		6		9	5	7		
9	1	4	6					
	2						3	7
8			5	1	2			4

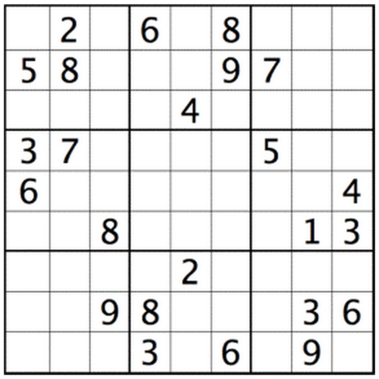


1	5	2	4	8	9	3	7	6
7	3	9	2	5	6	8	4	1
4	6	8	3	7	1	2	9	5
3	8	7	1	2	4	6	5	9
5	9	1	7	6	3	4	2	8
2	4	6	8	9	5	7	1	3
9	1	4	6	3	7	5	8	2
6	2	5	9	4	8	1	3	7
8	7	3	5	1	2	9	6	4

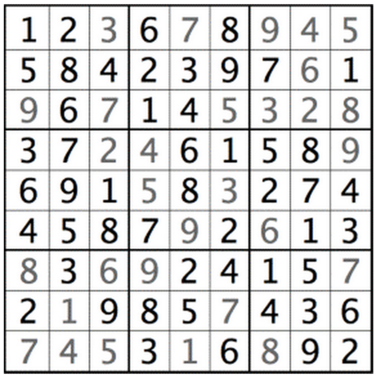




You can see as it gets very close to the solution, then decides that isnt it and gains chaos again, before settling down to a new state. Eventually, it must reach the point of highest potential, which is where we find the solution. This seems to work for easy puzzles and select others, but once we pick a difficult puzzle, the amount of iterations it would take seem to exceed a reasonable amount of computation. For example, the following puzzle:

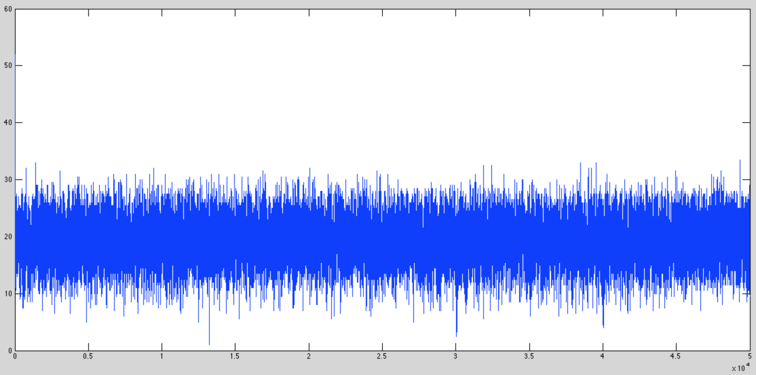


	2		6		8			
5	8				9	7		
				4				
3	7					5		
6								4
		8					1	3
				2				
		9	8				3	6
			3		6		9	



1	2	3	6	7	8	9	4	5
5	8	4	2	3	9	7	6	1
9	6	7	1	4	5	3	2	8
3	7	2	4	6	1	5	8	9
6	9	1	5	8	3	2	7	4
4	5	8	7	9	2	6	1	3
8	3	6	9	2	4	1	5	7
2	1	9	8	5	7	4	3	6
7	4	5	3	1	6	8	9	2

Will not solve in 50000 iterations. The potential can be seen below



It gets close sometimes, a value of 2 or 3. But we never reach the equilibrium. I think that I may be able to alter LLL to come up with a method that will solve any sudku puzzle. Perhaps changing T with time, or maybe a different method of chosing the player that will chose his move. In any case, I think this problem can be solved.

What is intreging is that it either takes less than 200-300 iterations, or it takes more than 50000. There must be a reason behind this.