Exploring the Use of Fuzzy Constraint Satisfaction Problems to Evaluate the Happiness of Society. Computer Systems Lab, 2009-2010

Peter Ballen

May 26, 2010

Abstract

The goal of this project is to explore the use of and solving of fuzzy constraint satisfaction. In this project, I will apply the prinicpals of fuzzy constraint satisfaction to a randomly generated society with the goal of making the digital populace as happy as possible.
Note: I have images, but am having difficulty getting them into Latex. Just assume you can see the pretty colorful images.

Keywords: fuzzy constraint satisfaction, soft constraint satisfaction.

1 Introduction - Elaboration on the problem statement, purpose, and project scope

1.1 Scope of Study

Fuzzy constraint satisfaction problems are similar to regular constraint satisfaction problems, but are far more useful in the real world. Regular constraint satisfaction problems are useful when all constraints are hard and cannot be violated, however they are only capable of finding a perfect solution. If no solution exists, the algorithm will fail. Fuzzy constraint satisfaction problesn

are used instead when the constraints are soft. Instead of demanding a perfect solution, it instead searches for an optimal solution that best satisfies the given constraints.

This is incredibly important in the real world because many real problems do not have simple "perfect" solutions. Instead, it becomes necessary to compromise and come up with the best answer. The work done in this project is a simple demonstration of applying this idea to a easily modeled society.

1.2 Type of research

The ultimate goal of this project is not to make a group of imaginary citizens "happy", but to explore the applications and extensions of fuzzy constraint satisfaction problems. The simulated society is useful in this regards as it posesses sufficient complexity to be nontrivial while being simple enough to easily model.

2 Background and review of current literature and research

To understand soft constraint satisfaction problems, it is essential to understand hard constraint satisfaction problems. The basis of any constraint satisfaction problem, hard of soft, is that there are a number of constraints, formally called tuples, that can be either satisfied or violated.

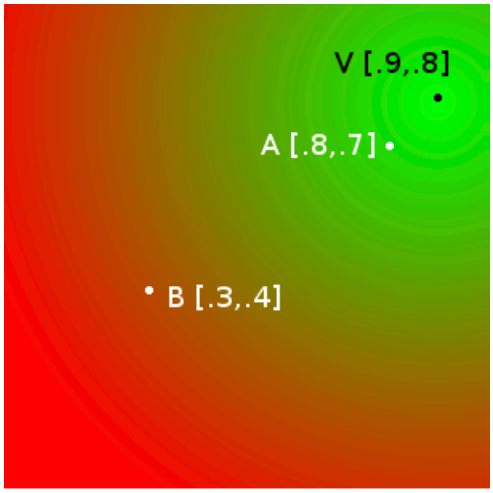
In a hard constraint satisfaction problem, all constraints are considered imperative and inflexible. This means that every tuple can only have two values, satisfied or violated. Furthermore, a solution is valid if and only if it satisfies every tuple. Examples of hard constraint satisfaction problems include Sudoku, the 4-Color Map Problem, and N-Queens. However, as will soon be aparent, this technique is not appropriate for many real world applications. Life often does not give easy answers with perfect solutions.

3 Procedures and Methodology

3.1 Society Model

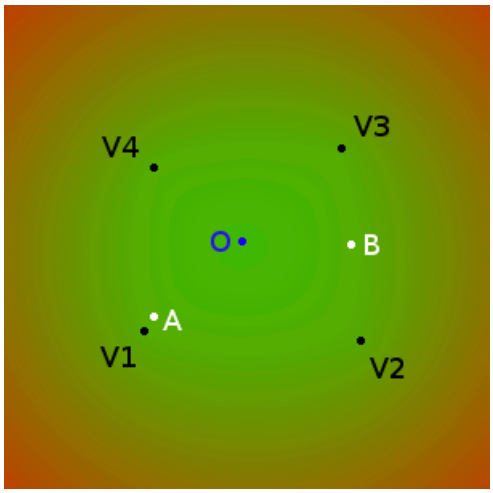
In order to further explore fuzzy constraint satisfaction, it becomes necessary to create a model of a simplified society. In this society, "Voters" (tuples) are randomly placed on a 1x1 board. A proposal is then placed somewhere on the board. Each Voter's satisfaction is given a value between 0 and 1, determined by a function dependent on the distance between that Voter and the Proposal. Society's satisfaction is calculated as the average satisfaction of each Voter.

For example, imagine for a moment a society with a single Voter, drawn in black at [.9,.8]. The background has been colored to correspond to society's satisfaction, the spots in green represent locations where the most constraints are satisfied and society is happy. By contrast, the spots in red represent locations where the most constraints are violated and society is unahppy. Proposal A [.8,.7] and Proposal B [.3,.4] are both drawn as white dots. Proposal A is closer to Voter V than Prospoal B; as such societies satisfaction is much higher at Point A than Point B.



We now move onto a more complicated example. Image a new society with four Voters, V1, V2, V3, and V4. It is possible to consider multiple scenarios that would make an individual voter happy. For example, Proposal A would make Voter V1 satisfied, but Voters V2, V3, and V4 are much less satisfied. Proposal B would make voters V2 and V3 happy, but Voters V1 and V4 are

unsatisfied. Ultimately, the optimal proposal, O, is a locaion in the center of the four dots. The imnportant piece of this model is that the optimal solution can only be obtained when every Voter sacrifices a bit of its personal satisfaction to best satisfy the society as a whole.

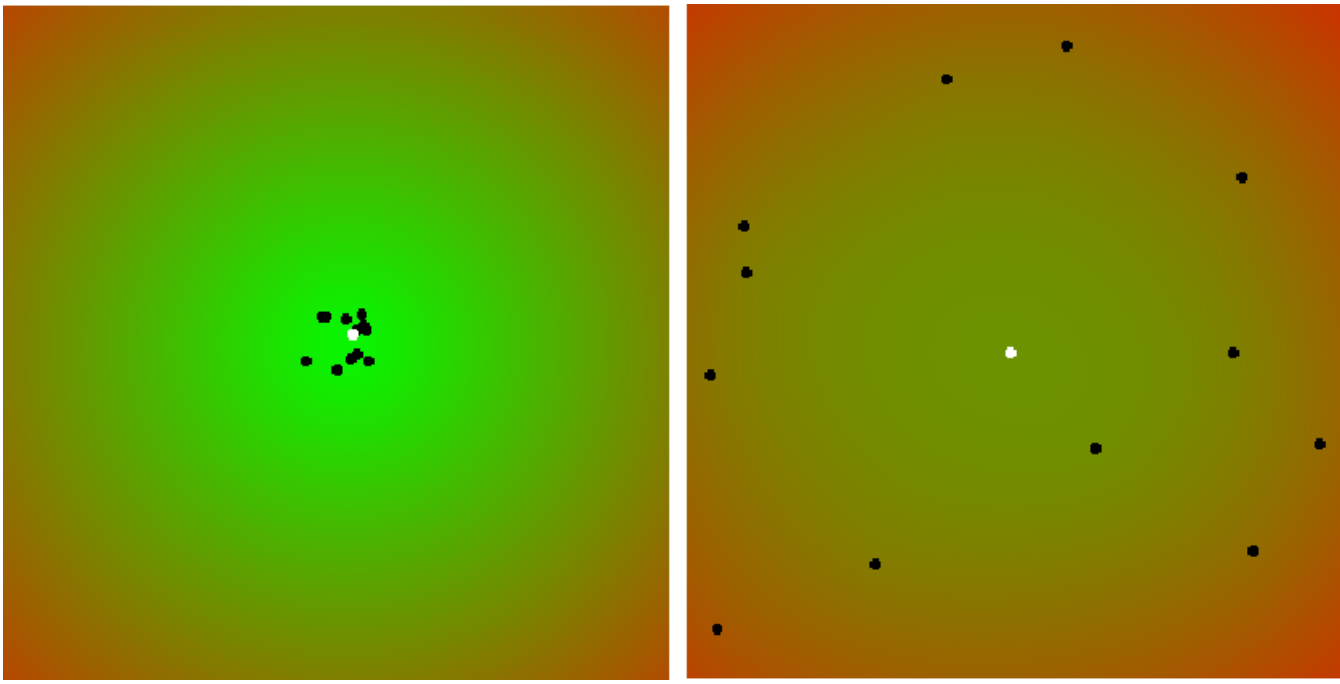


3.2 Sympathetic and Antagonistic Constraints

One interesting discussion is the idea of sympathetic and antagonistic constraints. Sympathetic constraints are two constraints located such that increasing the satisfaction of one by necessity increases the satisfaction of the other. Antagonistic constraints are just the opposite, increasing the satisfaction of one decreases the satisfaction of another. In the society model, Voters that are close to one another are sympathetic and Voters that are farther apart are antagonistic.

It is possible to force a society to be sympathetic or antagonistic by controlling the range the Voters are allowed to be placed in. For example, forcing the voters into a .1x.1 square centered at [.5,.5] will create a very sympathetic society, while randomly placing the Voters throughout the board will creatre a very antagonistic society.

As expected, as Society becomes more antagonistic and the Voters become more dispersed, the average satisfaction of society decreases. The very sympathetic society has an average satisfaction of 96 percent. In comparison, the very antagonistic society has an average satisfaction of 57 percent. It is important to note that, even in the incredibly sympathetic society, a perfect solution of 100 Satisfaction cannot be found.



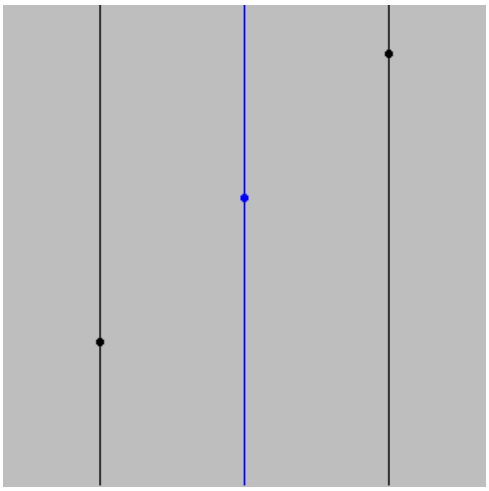
3.3 Finding the Solution

To this point, I have been generating solutions using a brute force method. Brute force is the idea that if you don't know how to find the solution, you just try everythinng. The code will look at every single point, evaluate the society's satisfaction at that point, then move onto the next point. While effective, this method is restrictively slow. For most real world applications, solutions must be generated in a reasonable timeframe that brute force simply cannot provide.

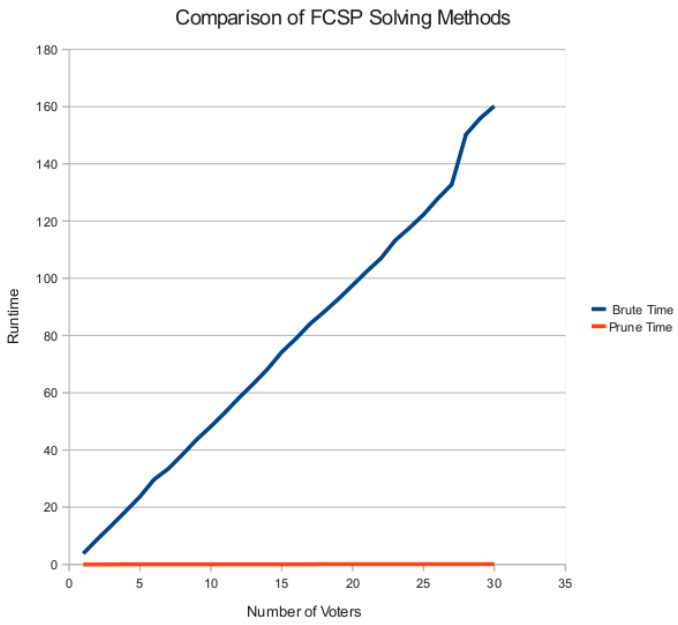
Thankfully, there are more effective methods to find a solution. One of these solutions is to implement pruning. Pruning is the idea that by ignoring solutions that are unlikely to produce the optimal solutions, the code can run in exponentially less time

In the Society model, pruning is accomplished by breaking the points into seperate single values, the x-value and the y-value. First, ignore the y-values and look only at the x-values; in the image, this is represented by turning the black dots into black lines. Second, find the optimal x-value using these results; in the image, this is represented by drawing in the blue line. Third,

prune away all possible choices that do not have this x-value; in the image, this is represented by the blue line and greying out the rest of the board. Finally, look at the remaining results to find the true optimal solution; in the image, this is represented by the blue dot.



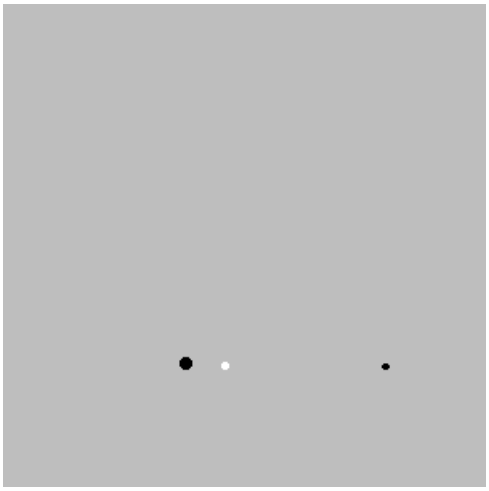
The runtime reduction of the pruning method is substancial. For example, solving a society of fifteen voters with brute force took 30 seconds, whereas pruning took 0.15 seconds. The graph above displays this well. Notice that as the number of Voters in the population increases, brute force gets slower and slower while pruning remains relatively constant. This holds true even with large populations. A society of one hundred thousand voters was solved in 42 seconds with pruning, it would take months to solve with brute force.



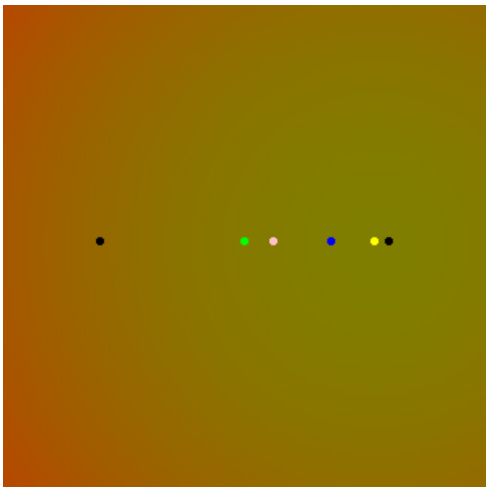
3.4 Prioritizing Constraints

So far, we have assumed that all of the constraints are of equal importance. In the society model, this corresponds to a fundamental principal of democracy: one man, one vote. However, in the real world, many constraints are considered more, or less, important than others. In order to better reflect this, we further modify the model.

We first assign each voter a priority level, a number between 0 and 1. Higher priorities are considered more important than lower priorities, and the total sum of the voters' priority is one. For example, consider a society with only two voters. The voter on the left has a priority level of 0.8 while the voter on the right has a priority level of 0.2. In other words, the voter on the left (the large black dot) is four times more important than the voter on the right (the small black dot). Notice that if the voter's were weighted equally, the optimal solution (the white dot) would be located exactly in the center of the two dots. However, because the Voters are not weighted equally, the optimal solution is slanted towards the Voter on the right.



It is relatively simple to compare how changing the priority assigned to each Voter changes the model. Observe the image above. The green dot is the optimal solution if both Voters are given equal priority. The pink dot is the optimal solution if the right dot is given .6 priority. The blue dot is the optimal solution if the right dot is given .8 priority. Finally, the yellow dot is the optimal solution if the right dot ig given .95 priority. Unsurprisingly, as the right voter is given a greater priority, the optimal solution approaches that Voter.



4 Results - Case Study

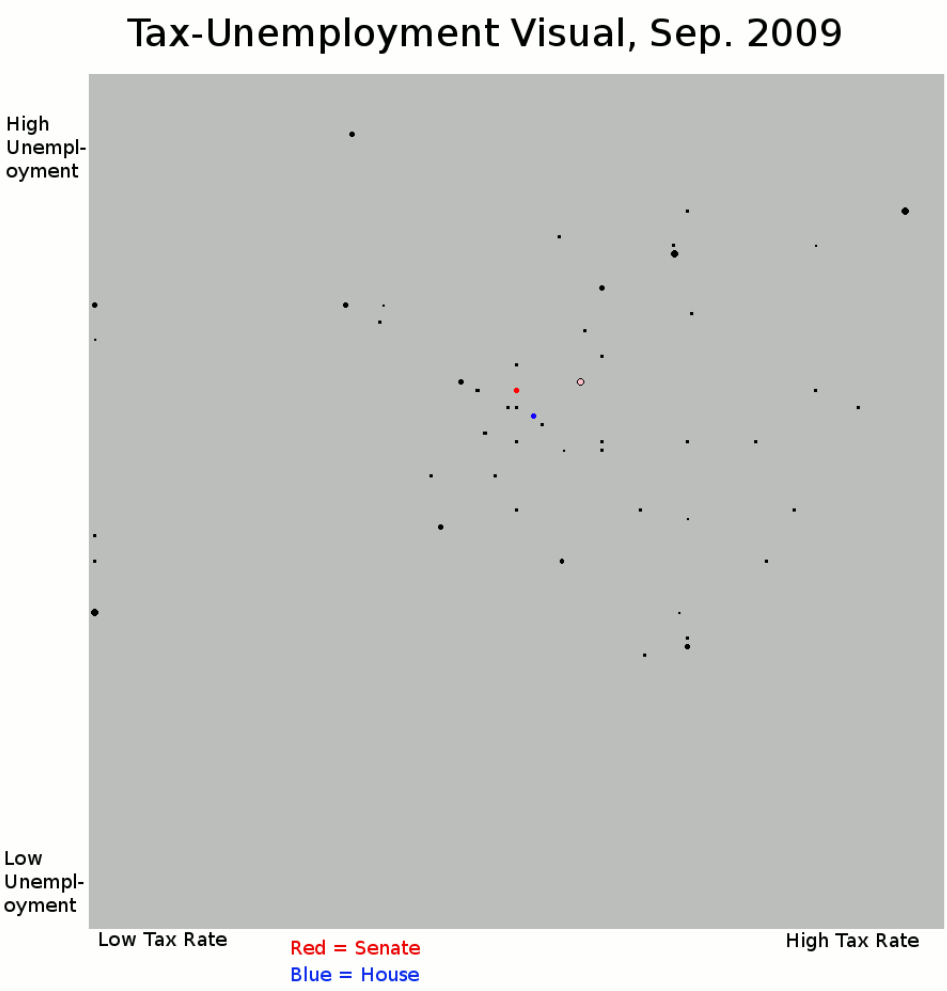
So far, the code has been used for demonstrative purposes - the Voters are voting on abstract proposals that don't mean anything in the real world. The following is a real-world example to help lend context and relevance to the ideas presented above.

Begin by thinking back to September, 2009. The United State's economy is in bad shape. The stock market has crashed, job creation is nonexistent, and the unemployment numbers are creeping higher and higher. An economic think-tank comes up with the following proposal.

In response to the failing economy, we must take drastic action. We propose that we substantially increase unemployment benefits (money given to unemployed men and women from the government to help them buy food and search for work). To offset the cost of these benefits, we recommend raising the national income tax on individuals and families making more than 50,000 dollars annually.

Wondering how the proposal would be received in Congress, the think-tank wants to predict public reaction to the plan. First, they gather information about the state income tax rate of each state (and the District of Columbia). Second, they gather information about the unemployment rate of each state. Using this data, it is possible to gain a rough predictor of how the Voters of that state would feel about the proposal.

Pennsylvania, for example, has a low income tax rate but a high unemployment rate. They would favor exxtending unemployment benefits, but they don't want to raise taxes to do so. Instead, they would favor deficit spending (funding the benefits by increasing the national debt). In contrast, South Carolina has a high tax rate but very low unemployment. They would be more inclined towards raising the income tax, but spending the increased revenue in other areas such as education or defense.



Using prioritized fuzzy constraint satisfaction, the think-tank calculated two numbers. The first number, drawn in blue, is the optimal proposal for the House of Representatives, where each states' representation is determined by its population. The second number, drawn in red, is the optiaml proposal for the Senate, where each state has an equal amount of representation.

Both numbers fall in the middle of the high/low tax rate axis, and slightly above the middle of the high/low unemployment rate axis. From this result, is becomes clear that both the House and the Senate would support a moderate tax level. They don't want too much deficit spending, but they also don't want to raise the income tax too high. Similarly, it becomes apparent that most states do favor slightly increasing unemployment benefits. The Senate

would prefer a larger bill that dips slightly into the deficit, with less taxes and more benefits. The House, in contrast, is in favor of a smaller bill funded only by tax money, at the cost of slightly less benefits.

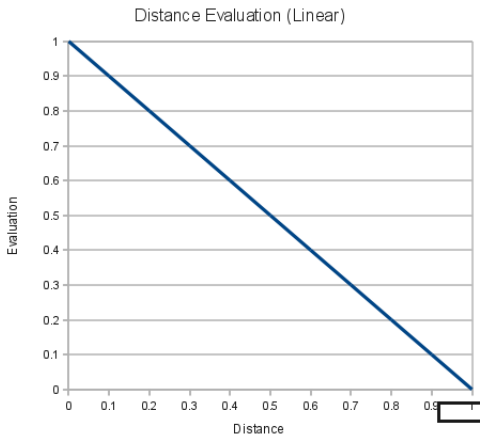
It can be intresting to manipulate the model to better understand how various constraints are connected to their states. The following commands are often helpful in such a task:

- Highlight - Highlight the dot that corresponds to a given state.
- Unhighlight - Unhighlight a previously highlighted dot.
- Poll - Give the happiness heuristic function of a given state.
- Identify - Give the opinion values of a given state.
- Mouse Input - Clicking by a dot will give the state it corresponds to.

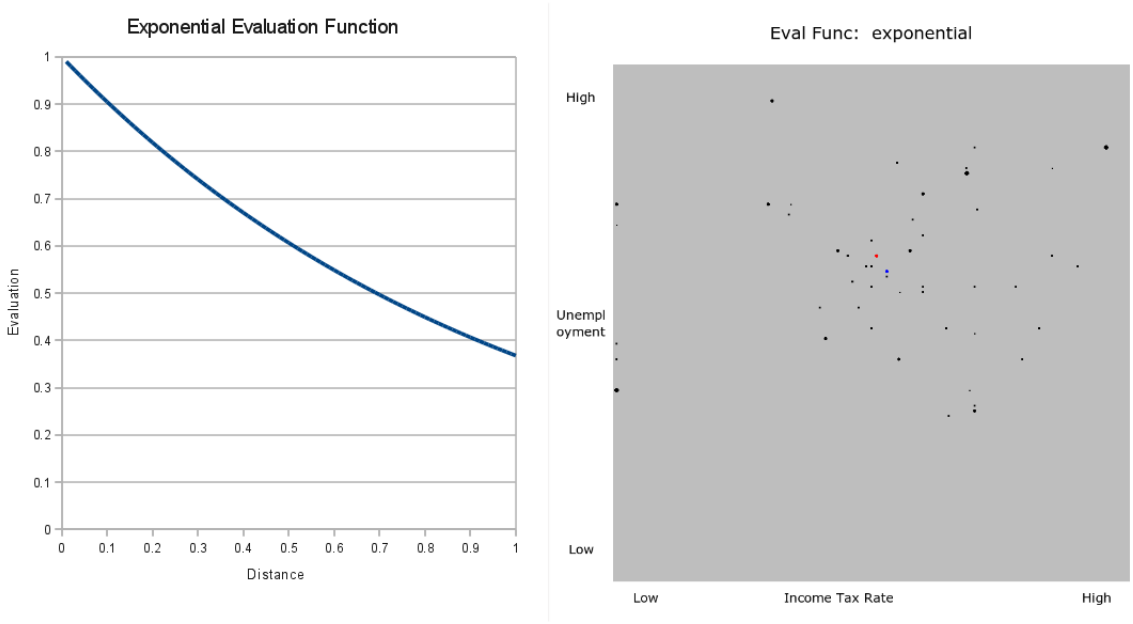
Using these commands, it is possible to gain a greater understanding of the model. It is difficult to replicate the interactivity in a paper, but hopefully the previous discussion has been clear enough to understand.

5 Extentions

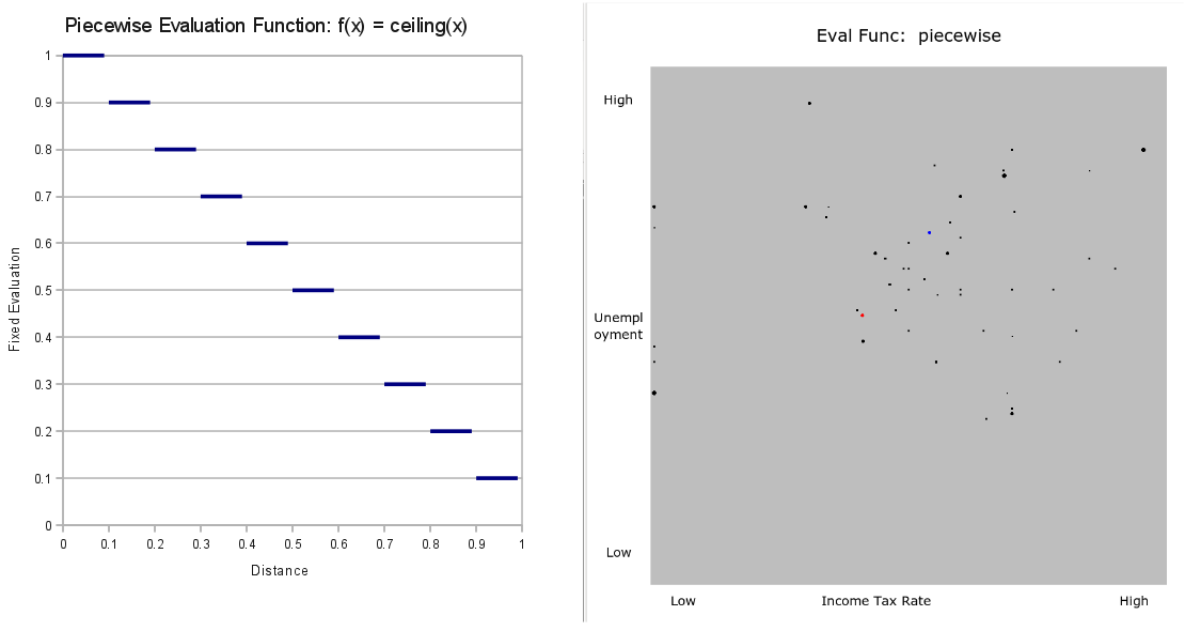
Throughout the Voter model, we have assumed that the best estimator of a Voter's happiness is a linear distance function, whose graph is shown below. With this function, given a distance d between a proposal and the Voter, h(d) = d.



The linear model was convenient because it was easy to visually interprete. However, there is no reason we could have not used a more complicated function. For example, consider the exponential function h(d) = exp(d)/exp(1), where d is still the distance between the Voter and the proposal. The graph and resulting model are shown below.



Notice how the optimal solutions, drawn in blue and red, have not dramatically change. It is also possible to consider a noncontinuous evaluation function. Consider the function h(d) = ceiling(d), where the ceiling command rounds the distance d up to the nearest ,1 increment. For example, h(.9) = h(.84) = h(.8001) = .9. The graph and resulting model are shown below.



In this case, the optimal solutions have varied dramatically: the weighted solution has moved towards the upper-right corner and the unweighted solution have shifted towards the bottom-left corner. The piecewise model shifts both solutions away from the middle towards the extremes.

These are all very simple extentions. More complicated extentions could use far more complicated equations, as well as having several more variables. However, no matter how complicated the model becomes, the fundamental ideas are identical to those presented earlier in the paper.

6 Conclusions

In this paper, we have discussed the fundamental differences between hard constraint satisfaction and soft constraint satisfacton. We have done basic constraint interpretation to classify constraints as antagonistic or sympathetic, and to prioritize constraints based on importance. Finally, we used these ideas to construct a basic model of taxes and unemployment and evaluated tax proposals. This paper has discussed the basic idea of fuzzy constraint satisfaction and its use in finding solutions to problems that do not have a perfect solution.

References

[1] Aron Armstrong, Edmund Durfee Dynamic Prioritization of Complex Agents in Distributed C University of Michigan

[2] Vipin Kumar, Agorithms for Constraint Satisfaction: A Survey, AI Magazine, 1992.

[3] Yasusi Kanada, Fuzzy Constraint Satisfaction Using CCM:, Tsukuba Research Center, 1995.

[4] Zs. Ruttkay, Fuzzy Constraint Satisfaction, Vrije University Amsterdam

[5] D Dubois, H Fargier, H Prade, Possibilistic Constraint Satisfaction Problems, Applied Inteligence, 1996

[6] Loretta Bass, Lynne Casper, Population Division Working Paper No. 28, U.S. Bureau of the Census, 1999

[7] Eugene Freuder, Partial Constraint Satisfaction, University of New Hampshire, 1992

[8] US Tax Foundation, State Individual Income Taxes 2009, 2009

[9] Bureau of Labor Statistics, Unemployment Rates for States, 2009