# Constraint Satisfaction Problem Solver

*Valid Everywhere*

## Tautology

- Becky Everson
- Thomas Hollowell
- Francis Zhou
- Tait Madsen
- Evan McLaughlin
- Tom Craw

## Executive Summary

Constraint Satisfaction Problem Solver (CSPS) is true to it's name, it solves constraint satisfaction problems. What then, is a constraint satisfaction problem? It is a problem that can be defined in terms of variables, domains for those variables, and constraints - the solution to which consists of an assignment of a domain to each variable in a way that satisfies all constraints.

Many problems can be generalized to constraint satisfaction problems, such as coloring a map so that no two adjacent regions are the same color, or choosing a book for a book club whose members have complicated preferences.

CSPS will provide both a user interface and an API to interact with its algorithm. This way both casual users and websites alike can benefit from the ability to solve complicated problems that fall within the scope of constraint satisfaction.

### Use Case 1: Family Matters

Congratulations, your aunt Marge is finally getting married! Unfortunately, you don't live in one, big, happy family… Since you are the maid of honor (or best man with maid of honor duties), you must make the seating chart for the reception. There are going to be 5 family members sitting at one table: cousin Jane, grandpa Al, uncle Joe, aunt Mary, and grandma Shirley. If you remember correctly, cousin Jane doesn't want to sit next to grandpa Al, uncle Joe doesn't like being next to his wife (aunt Mary) or his dad (grandpa Al), and neither uncle Joe nor grandpa Al wants to be next to grandma Shirley.

Using the Constraint Satisfaction Problem Solver, you whip up a seating chart in no time and aunt Marge has the wedding reception that she always dreamed of; one without family conflicts at the dinner table! Uncle Joe didn't even have too much wine!

### Use Case 2: Social Media

A social media site (SM) wants to add more functionality to its event-planning feature. A user who creates an event will naturally be able to invite anyone they want; however, perhaps the user wants to flesh out a gathering. The user could specify how many people they want to invite, to what degree should those people know each other (how many "friends" will each guest have who are also present).

This has now become a constraint satisfaction problem, so instead of implementing an algorithm themselves, SM chooses to use CSPS's API! Once SM gets the necessary information from its user, it sends the information to our server to be processed. Once the problem is solved, the information is sent back to SM so it can suggest a guest list to its user.

### Use Case 3: Sudoku

Everyone has heard of Sudoku! This popular game is also a constraint satisfaction problem. Unfortunately, Mike has been slaving over one Sudoku puzzle for days and is stuck. By entering the rules of Sudoku and the given puzzle into the CSPS user interface, Mike can have CSPS solve the puzzle for him.

### Use Case 4: Which Hat Should Tommy Buy?

Tommy is a great son and grandson. Tommy will visit his grandparent's during Christmas with his parents. Tommy wants to buy a hat to wear, but the color is hard to choose because people living in their family are so picky with the color. Tommy's grandpa, John, doesn't like red because John is Veteran and he doesn't like communism. Tommy's grandma, Mary, doesn't like black because she wants to live forever. Tommy's parents are so afraid of sea because they don't know how to swim, so they don't like blue. Tommy's younger sister don't like dark color because she is afraid of ghost. Tommy's the elder sister doesn't matter the color. Which color of hat should Tommy buy in order to make everyone happy? By entering the preferences of each of his family members into CSPS, Tommy can solve this conundrum.

### Required Technology

1. **Use a Server-Side Framework** - Will be using other technologies besides HTML/CSS, we will being using PHP and Ajax as well.
2. **Make an API** - Will have an API to the algorithm accessible to other websites.
3. **Design & Evaluate** - This will have a friendly user interface to maximize usability. This will be evaluated with test subjects.
4. **Accessibility** - Due to the complexity of many constraint satisfaction problems, it is imperative that our site streamline the process of entering necessary information.
5. **Ajax** - will allow client-side interaction to continue while the server process problem.