Constraint Satisfaction Problems

Constraint satisfaction problems (CSPs) are problems in which the search is for a **feasible** solution, meaning one that satisfies all the constraints. By definition, constraint satisfaction involves "hard" constraints—constraints that either pass or fail, with no middle ground. Nonetheless, constraint satisfaction problems are often solved as optimization problems with "soft" constraints. The objective is to maximize the number of constraints satisfied; or equivalently, to minimize the number of constraints violated. Examples of CSPs include cryptoarithmetic puzzles and crossword puzzles.

Formally, a (finite) **constraint satisfaction problem** is a triple $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, where

- a finite set of variables $\mathcal{X} = \{x_1, \ldots, x_n\}$
- a set of finite domains $\mathcal{D} = \{D_1, \ldots, D_n\}$
- a finite set of constraints

In finite CSPs, constraints can be expressed either *extensionally* or *intensionally*. Given a CSP with two variables x_1 and x_2, if $D_1 = \{A, B\}$ and $D_2 = \{B, C\}$, then the constraint "the value of x_1 cannot equal the value of x_2" is expressed extensionally as $\{(A, B), (A, C), (B, C)\}$ and intensionally as $x_1 \neq x_2$.

An *assignment* is a mapping from variables to values. A *complete* assignment assigns a value to all variables. A *consistent* assignment does not violate any constraints. A *solution* to a CSP is a complete and consistent assignment.

Example A classic example of a CSP is the n-queens problem. In this problem, n-queens are to be placed on an $n \times n$ chess board *s.t.* no two queens threaten each other: i.e., no two queens occupy the same row, column, or diagonal. A solution to the 8-queens problem is depicted in Figure 1.

				Q			
		Q					
Q							
						Q	
	Q						
							Q
					Q		
			Q				

Figure 1: A solution to the 8-queens problem.

In the case of 4-queens, the problem can be represented as follows.

- variables $x_{r1}, x_{c1}, x_{r2}, x_{c2}, x_{r3}, x_{c3}, x_{r4}, x_{c4}$, where variable x_{ri} represents the row of the i-th queen and variable x_{ci} represents the column of the i-th queen.
- domains: $D_{r1} = D_{c1} = D_{r2} = D_{c3} = D_{r3} = D_{c3} = D_{r4} = D_{c4} = \{1, 2, 3, 4\}$
- constraints:

- no two queens can occupy the same row: i.e., $x_{r1} \neq x_{r2}$, $x_{r1} \neq x_{r3}$, $x_{r1} \neq x_{r4}$, $x_{r2} \neq x_{r3}$, $x_{r2} \neq x_{r4}$, $x_{r3} \neq x_{r4}$
- no two queens can occupy the same column: this constraint is formulated in the same way as the row constraints, but using the column variables.
- no two queens can occupy the same diagonal: i.e.,
  * $|x_{r1} - x_{r2}| \neq |x_{c1} - x_{c2}|$
  * $|x_{r1} - x_{r3}| \neq |x_{c1} - x_{c3}|$
  * $|x_{r1} - x_{r4}| \neq |x_{c1} - x_{c4}|$
  * $|x_{r2} - x_{r3}| \neq |x_{c2} - x_{c3}|$
  * $|x_{r2} - x_{r4}| \neq |x_{c2} - x_{c4}|$
  * $|x_{r3} - x_{r4}| \neq |x_{c3} - x_{c4}|$

The assignment $\{x_{r1} \mapsto 3, x_{c1} \mapsto 1, x_{r2} \mapsto 1, x_{c2} \mapsto 2, x_{r3} \mapsto 4, x_{c3} \mapsto 3, x_{r4} \mapsto 2, x_{c4} \mapsto 4\}$ solves the 4-queens problem.

One approach to solving CSPs is to use blind search methods that have been extended with general-purpose heuristics suited to CSPs. We propose a different technique in these notes: reduce an instance of a CSP to an instance of SAT, and then run a SAT solver in search of a solution to the CSP. We demonstrate this approach on n-queens.

To start, we need to define our variables. Note that the variables in the aforementioned formulation of the problem are integer-valued, not boolean-valued. Therefore, they are not suitable for formulating n-queens as SAT. A popular alternative is to use a **one-hot encoding**, which means creating a boolean variable for each possible integer value. In other words, we could create boolean variables x_{ij}, where the value 1 indicates that a queen in the ith column is in the jth row.

Having selected our variables, we now move on to formulating the constraints.

Each row must have exactly one queen. We formulate this constraint as the conjunction of multiple constraints, namely there must be at least one queen in each row, but there cannot be two (or more) queens in the same row.

- **At least one queen in each row.**

  For all rows $1 \leq i \leq n$, $(x_{i,1} \vee x_{i,2} \ldots \vee x_{i,n})$.

  How many such constraints are there? There are n of these constraints, since there are n rows.

- **No two queens in the same row.** For all rows $1 \leq i \leq n$ and all pairs of distinct columns $1 \leq j \neq k \leq n$, $(\neg x_{i,j} \vee \neg x_{i,k})$.

  How many such constraints are there? There are $n \times n \times n - 1$ such constraints, since there are n rows and there are $n \times n - 1$ pairs of distinct columns in each row.

Each column must have exactly one queen. This constraint is analogous to the previous constraint, but we implement it by iterating over columns instead of rows.

- **At least one queen in each column**

  For all columns $1 \leq j \leq n$, $(x_{1,j} \vee x_{2,j} \vee \ldots \vee x_{n,j})$.

- **No two queens in the same column**

  For all columns $1 \leq j \leq n$ and all pairs of distinct rows $1 \leq i \neq k \leq n$, $(\neg x_{i,j} \vee \neg x_{k,j})$.

Each diagonal must have exactly one queen. To represent this constraint, we consider both the main diagonals (top-left to bottom-right) and the anti-diagonals (top-right to bottom-left).

Main diagonals For all pairs (i, j) and (k, l) where $i - j = k - l$ and $i \neq k$, $(\neg x_{i,j} \vee \neg x_{k,l})$.

Anti-diagonals For all pairs (i, j) and (k, l) where $i + j = k + l$ and $i \neq k$, $(\neg x_{i,j} \vee \neg x_{k,l})$.

If we can find an assignment that satisfies the conjunction of all these clauses, then we will have found a solution to the n-queens problem!