Constraint Satisfaction– Solution

1) Consider the two formulations of the N-Queens problem as a constraint satisfaction problem from the slide set. Compare these two formulations in terms of the size and branching factor of the state space and the depth of the search tree.

Answer:
In the first formulation, there are N^2 variables, each with domain $\{0, 1\}$. Therefore, the size of the state space is 2^{N^2} (note that this is not the number of valid configurations). Since we can pick from at most two values from the domain of a variable, the branching factor is 2. Since we have to assign values to N^2 variables, the depth of the search tree is N^2.

In the second formulation, there are N variables, each with domain $\{1, \ldots, N\}$. Therefore, the size of the state space is only N^N. Since we can pick from at most N different values from the domain of a variable, the branching factor is N. Since we have to assign values to N variables, the depth of the search tree is N.

2) In the crossword puzzle, we have a grid with blocked and unblocked cells and a dictionary of words. We want to assign a letter to each unblocked cell so that each vertical or horizontal contiguous segment of unblocked cells form a word that appears in the dictionary. An example of a solved crossword puzzle is given below[1].

A	D	I	M	■	R	I	P	S	■	F	A	T
C	U	T	E	■	E	T	A	T	■	A	L	E
D	E	S	D	E	M	O	N	A	■	I	A	N
C	L	A	U	D	I	O	■	T	U	R	N	S
■	■	■	S	E	T	■	T	E	R	M	■	■
A	S	W	A	N	■	P	E	N	N	A	M	E
D	I	I	■	■	H	A	L	■	■	I	M	S
O	R	L	A	N	D	O	■	E	D	D	I	E
■	■	D	I	A	S	■	A	S	U	■	■	■
S	T	O	M	P	■	P	R	A	N	C	E	D
E	R	A	■	P	E	T	R	U	C	H	I	O
L	I	T	■	E	L	S	A	■	A	I	N	T
L	O	S	■	R	I	D	S	■	N	A	S	H

[1]http://www.americanshakespearecenter.com/v.php?pg=684

Formulate this puzzle as a constraint satisfaction problem. Describe the variables, their domains and the constraints. (Bonus question: Try to come up with a second formulation of this puzzle as a constraint satisfaction problem.)

Answer:
Formulation 1:

Variables: Unblocked cells.

Domains: The domain of each variable is the set of letters of the alphabet.

Constraints: For each vertical or horizontal contiguous segment of unblocked cells, we add a constraint between the cells in that segment, constraining the letters assigned to the cells in that segment to form a word that appears in the dictionary.

Formulation 2:

Variables: All vertical or horizontal contiguous segments of unblocked cells.

Domains: The domain of each variable is the set of words in the dictionary of the same length as the corresponding contiguous segment.

Constraints: For each pair of vertical and horizontal contiguous segments of unblocked cells that intersect at an unblocked cell s, we add a constraint between them, constraining the words assigned to them to have the same letter at s.

3) Consider the following constraint satisfaction problem with variables x, y and z, each with domain $\{1, 2, 3\}$, and constraints C_1 and C_2, defined as follows:

- C_1 is defined between x and y and allows the pairs $(1, 1), (2, 2), (3, 1), (3, 2), (3, 3)$. (A pair (a, b) means that assigning $x = a$ and $y = b$ does not violate the constraint. Any assignment that does not appear in the list violate the constraint.)
- C_2 is defined between y and z and allows the pairs $(1, 1), (1, 2), (3, 1), (3, 2), (3, 3)$.

Which values does arc consistency rule out from the domain of each variable? Suppose that we started search after establishing arc consistency, and we assign $x = 1$. Which values does a) forward checking and b) maintaining arc consistency rule out from the domain of each variable?

Answer:
Arc consistency first rules out $y = 2$ because there is no value of z that satisfies C_2 if $y = 2$. It then rules out $x = 2$ because, once $y = 2$ is ruled out, there is no value of y that satisfies C_1 if $x = 2$. Arc consistency does not rule out any other values at this point.

If we assign $x = 1$ during search, forward checking rules out only $y = 3$ ($y = 2$ is already ruled out by arc consistency). Maintaining arc consistency rules out $y = 3$ and $z = 3$.

4) Suppose you have a search problem defined by more or less the usual stuff:

- a set of states S;
- an initial state s_{start};
- a set of actions A including the *NoOp* action, that has no effect;
- a transition model $Result(s, a)$ (that determines the successor state when action a is executed in state s);
- a set of goal states G.

Unfortunately, you have no search algorithms! All you have is a CSP solver.

(a) Given some time horizon T, explain how to formulate a CSP such that (1) the CSP has a solution exactly when the problem has a solution of length T steps; (2) the solution to the original problem can be "read off" from the variables assigned in CSP solution. Your formulation must give the variables, their domains, and all applicable constraints expressed as precisely as possible. You should have at least one variable per time step, and the constraints should constrain the initial state, the final state, and consecutive states along the way.

(b) Explain how to modify your CSP formulation so that the CSP has a solution when the problem has a solution of length $\leq T$ steps, rather than exactly T steps.

Answer:
We use the variables $s_0 \dots s_T$ and $a_0, \dots a_{T-1}$. The domain of s_0 is $\{s_{start}\}$, the domain of s_T is G, the domains of $s_1, \dots s_{T-1}$ are S, and the domains of $a_0, \dots a_{T-1}$ are A. For each $i = 0 \dots T-1$, we have the following constraint: $Result(s_i, a_i) = s_{i+1}$, that is, the execution of action a_i in state s_i results in state s_{i+1} (and $a_i \in A(s_i)$, where $A(s_i)$ is the set of actions that can be executed in s_i). If we include the *NoOp* action in A (and all $A(s_i)$), we find a solution of length $\leq T$ steps. Otherwise, we find a solution of length T steps. In both cases, the solution can be read off from the assignments to the variables $a_0 \dots a_{T-1}$.

5) Show how a single ternary constraint such as "A + B = C" can be turned into three binary constraints by using an auxiliary variable. You may assume finite domains. (Hint: Consider a new variable that takes on values that are pairs of other values, and consider constraints such as "X is the first element of the pair Y.") Next, show how constraints with more than three variables can be treated similarly. Finally, show how unary constraints can be eliminated by altering the domains of variables. This completes the demonstration that any CSP can be transformed into a CSP with only binary constraints.

Answer:
To turn a single constraint C between n variables $x_1, \dots, x_n$ into n binary constraints, we can add a new variable y whose domain is a subset of the

Cartesian product of the n variables' domains. Namely, this subset contains exactly the tuples $< a_1, \dots a_n >$ where, $\forall i \in \{1 \dots n\}$, a_i is in the domain of x_i, and the assignment $x_1 = a_1, \dots, x_n = a_n$ is not ruled out by C. We then add the binary constraints $B_1, \dots, B_n$ where, $\forall i \in \{1 \dots n\}$, B_i is a constraint between x_i and y that rules out any assignment $x_i = a_i$, $y =< b_1, \dots, b_n >$ if and only if $a_i \neq b_i$.

A unary constraint is a constraint over a single variable, ruling out some of the values from its domain. Instead of using a unary constraint to rule out values from the domain of a variable, we can simply remove those values from the variable's domain.