EXERCISES 5

CONSTRAINT SATISFACTION PROBLEMS

5.1 Defining Constraint Satisfaction Problems

Exercise 5.AUSM

How many solutions are there for the three-color map-coloring problem in Figure 5.1? How many solutions if four colors are allowed? Two colors?

There are 18 solutions for coloring Australia with three colors. Start with *SA*, which can have any of three colors. Then moving clockwise, *WA* can have either of the other two colors, and everything else is strictly determined; that makes 6 possibilities for the mainland, times 3 for Tasmania yields 18. There are no solutions with 2 colors, because there are several places where three territories all neighbor each other. There are 768 solutions with four colors, as demonstrated by the following program (which also verifies the other counts with the result: 3: 18, 4: 768, 2: 0).

```
from itertools import product

def valid(WA, NT, SA, Q, NSW, V, T) -> bool:
    """Is this assignment of colors to territories valid?"""
    return (SA not in {WA, NT, Q, NSW, V}
            and WA != NT != Q != NSW != V)

{N: sum(valid(*colors) for colors in product(range(N), repeat=7))
 for N in (3, 4, 2)}
```

Exercise 5.MDBS

Mom, Dad, Baby, Student, Teacher, and Guide are lining up next to each other in six linear spots labeled 1 to 6, one to a spots. Baby needs to line up between Mom and Dad. Student and Teacher need to be next to each other. Guide needs to be at one end, in spot 1 or 6. Formulate this problem as a CSP: list the variables, their domains, and the constraints. Encode unary constraints as a constraint rather than pruning the domain. (No need to solve the problem, just provide variables, domains and constraints.)

- Variables: $\{M, D, B, S, T, G\}$
- Domains: $\{1, 2, 3, 4, 5, 6\}$ for all variables
- Constraints: $\text{alldiff}(M, D, B, S, T, G), |B - M| = 1, |B - P| = 1, |S - T| = 1, G \in \{1, 6\}$.

Exercise 5.KKNI

Consider the problem of placing k knights on an $n \times n$ chessboard such that no two knights are attacking each other, where k is given and $k \leq n^2$.

a. Choose a CSP formulation. In your formulation, what are the variables?

b. What are the possible values of each variable?

c. What sets of variables are constrained, and how?

d. Now consider the problem of putting *as many knights as possible* on the board without any attacks. Explain how to solve this with local search by defining appropriate ACTIONS and RESULT functions and a sensible objective function.

a. Solution A: There is a variable corresponding to each of the n^2 positions on the board.
Solution B: There is a variable corresponding to each knight.

b. Solution A: Each variable can take one of two values, {occupied,vacant}
Solution B: Each variable's domain is the set of squares.

c. Solution A: every pair of squares separated by a knight's move is constrained, such that both cannot be occupied. Furthermore, the entire set of squares is constrained, such that the total number of occupied squares should be k.
Solution B: every pair of knights is constrained, such that no two knights can be on the same square or on squares separated by a knight's move. Solution B may be preferable because there is no global constraint, although Solution A has the smaller state space when k is large.

d. Any solution must describe a *complete-state* formulation because we are using a local search algorithm. For simulated annealing, the successor function must completely connect the space; for random-restart, the goal state must be reachable by hillclimbing from some initial state. Two basic classes of solutions are:
Solution C: ensure no attacks at any time. Actions are to remove any knight, add a knight in any unattacked square, or move a knight to any unattacked square.
Solution D: allow attacks but try to get rid of them. Actions are to remove any knight, add a knight in any square, or move a knight to any square.

Exercise 5.XWRD

Consider the problem of constructing (not solving) crossword puzzles: fitting words into a rectangular grid. The grid, which is given as part of the problem, specifies which squares are blank and which are shaded. Assume that a list of words (i.e., a dictionary) is provided

and that the task is to fill in the blank squares by using any subset of the word list. Formulate this problem precisely in two ways:

a. As a general search problem. Choose an appropriate search algorithm and specify a heuristic function. Is it better to fill in blanks one letter at a time or one word at a time?

b. As a constraint satisfaction problem. Should the variables be words or letters?

Which formulation do you think will be better? Why?

a. Crossword puzzle construction can be solved many ways. One simple choice is depth-first search. Each successor fills in a word in the puzzle with one of the words in the dictionary. It is better to go one word at a time rather than one letter at a time, to minimize the number of steps.

b. As a CSP, there are even more choices. You could have a variable for each box in the crossword puzzle; in this case the value of each variable is a letter, and the constraints are that the letters must make words. This approach is feasible with a most-constraining value heuristic. Alternately, we could have each string of consecutive horizontal or vertical boxes be a single variable, and the domain of the variables be words in the dictionary of the right length. The constraints would say that two intersecting words must have the same letter in the intersecting box. Solving a problem in this formulation requires fewer steps, but the domains are larger (assuming a big dictionary) and there are fewer constraints. Both formulations are feasible.

Exercise 5.CSPD

Give precise formulations for each of the following as constraint satisfaction problems:

a. Rectilinear floor-planning: find non-overlapping places in a large rectangle for a number of smaller rectangles.

b. Class scheduling: There is a fixed number of professors and classrooms, a list of classes to be offered, and a list of possible time slots for classes. Each professor has a set of classes that they can teach.

c. Hamiltonian tour: given a network of cities connected by roads, choose an order to visit all cities in a country without repeating any.

a. For rectilinear floor-planning, one possibility is to have a variable for each of the small rectangles, with the value of each variable being a 4-tuple consisting of the x and y coordinates of the upper left and lower right corners of the place where the rectangle will be located. The domain of each variable is the set of 4-tuples that are the right size for the corresponding small rectangle and that fit within the large rectangle. Constraints say that no two rectangles can overlap; for example if the value of variable R_1 is $[0, 0, 5, 8]$, then no other variable can take on a value that overlaps with the $0, 0$ to $5, 8$ rectangle.

b. For class scheduling, one possibility is to have three variables for each class, one with times for values (e.g. MWF8:00, TuTh8:00, MWF9:00, ...), one with classrooms for values

(e.g. Wheeler110, Evans330, ...) and one with instructors for values (e.g. Abelson, Bibel, Canny, ...). Constraints say that only one class can be in the same classroom at the same time, and an instructor can only teach one class at a time. There may be other constraints as well (e.g. an instructor should not have two consecutive classes).

c. For Hamiltonian tour, one possibility is to have one variable for each stop on the tour, with binary constraints requiring neighboring cities to be connected by roads, and an AllDiff constraint that all variables have a different value.

Exercise 5.CRYP

Solve the cryptarithmetic problem in Figure 5.2 by hand, using the strategy of backtracking with forward checking and the MRV and least-constraining-value heuristics. Show the trace of each variable choice and assignment.

The exact steps depend on certain choices you are free to make; here are the ones I made:

a. Choose the X_3 variable. Its domain is $\{0, 1\}$.

b. Choose the value 1 for X_3. (We can't choose 0; it wouldn't survive forward checking, because it would force F to be 0, and the leading digit of the sum must be non-zero.)

c. Choose F, because it has only one remaining value.

d. Choose the value 1 for F.

e. Now X_2 and $\mathbf{X}_1$ are tied for minimum remaining values at 2; let's choose X_2.

f. Either value survives forward checking, let's choose 0 for X_2.

g. Now X_1 has the minimum remaining values.

h. Again, arbitrarily choose 0 for the value of X_1.

i. The variable O must be an even number (because it is the sum of $T + T$ less than 5 (because $O + O = R + 10 \times 0$). That makes it most constrained.

j. Arbitrarily choose 4 as the value of O.

k. R now has only 1 remaining value.

l. Choose the value 8 for R.

m. T now has only 1 remaining value.

n. Choose the value 7 for T.

o. U must be an even number less than 9; choose U.

p. The only value for U that survives forward checking is 6.

q. The only variable left is W.

r. The only value left for W is 3.

s. This is a solution.

This is a rather easy (under-constrained) puzzle, so it is not surprising that we arrive at a solution with no backtracking (given that we are allowed to use forward checking).

Exercise 5.NARY

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.

The problem statement sets out the solution fairly completely. To express the ternary constraint on A, B and C that $A + B = C$, we first introduce a new variable, AB. If the domain of A and B is the set of numbers N, then the domain of AB is the set of pairs of numbers from N, i.e. $N \times N$. Now there are three binary constraints, one between A and AB saying that the value of A must be equal to the first element of the pair-value of AB; one between B and AB saying that the value of B must equal the second element of the value of AB; and finally one that says that the sum of the pair of numbers that is the value of AB must equal the value of C. All other ternary constraints can be handled similarly.

Now that we can reduce a ternary constraint into binary constraints, we can reduce a 4-ary constraint on variables A, B, C, D by first reducing A, B, C to binary constraints as shown above, then adding back D in a ternary constraint with AB and C, and then reducing this ternary constraint to binary by introducing CD.

By induction, we can reduce any n-ary constraint to an $(n-1)$-ary constraint. We can stop at binary, because any unary constraint can be dropped, simply by moving the effects of the constraint into the domain of the variable.

Exercise 5.ZEBR

Consider the following logic puzzle: In five houses, each with a different color, live five persons of different nationalities, each of whom prefers a different brand of candy, a different drink, and a different pet. Given the following facts, the questions to answer are "Where does the zebra live, and in which house do they drink water?"

The Englishman lives in the red house.
The Spaniard owns the dog.
The Norwegian lives in the first house on the left.
The green house is immediately to the right of the ivory house.
The man who eats Hershey bars lives in the house next to the man with the fox.
Kit Kats are eaten in the yellow house.
The Norwegian lives next to the blue house.
The Smarties eater owns snails.
The Snickers eater drinks orange juice.
The Ukrainian drinks tea.
The Japanese eats Milky Ways.

Kit Kats are eaten in a house next to the house where the horse is kept.
Coffee is drunk in the green house.
Milk is drunk in the middle house.

Discuss different representations of this problem as a CSP. Why would one prefer one representation over another?

The "Zebra Puzzle" can be represented as a CSP by introducing a variable for each color, pet, drink, country, and cigarette brand (a total of 25 variables). The value of each variable is a number from 1 to 5 indicating the house number. This is a good representation because it easy to represent all the constraints given in the problem definition this way. (We have done so in the Python implementation of the code.) Besides ease of expressing a problem, the other reason to choose a representation is the efficiency of finding a solution. here we have mixed results—on some runs, min-conflicts local search finds a solution for this problem in seconds, while on other runs it fails to find a solution after minutes.

Another representation is to have five variables for each house, one with the domain of colors, one with pets, and so on.

5.2 Constraint Propagation: Inference in CSPs

Exercise 5.GRAE

Consider the graph with 8 nodes A_1, A_2, A_3, A_4, H, T, F_1, F_2. A_i is connected to A_{i+1} for all i, each A_i is connected to H, H is connected to T, and T is connected to each F_i. Find a 3-coloring of this graph by hand using the following strategy: backtracking with conflict-directed backjumping, the variable order A_1, H, A_4, F_1, A_2, F_2, A_3, T, and the value order R, G, B.

a. $A_1 = R$.

b. $H = R$ conflicts with A_1.

c. $H = G$.

d. $A_4 = R$.

e. $F_1 = R$.

f. $A_2 = R$ conflicts with A_1, $A_2 = G$ conflicts with H, so $A_2 = B$.

g. $F_2 = R$.

h. $A_3 = R$ conflicts with A_4, $A_3 = G$ conflicts with H, $A_3 = B$ conflicts with A_2, so backtrack. Conflict set is $\{A_2, H, A_4\}$, so jump to A_2. Add $\{H, A_4\}$ to A_2's conflict set.

i. A_2 has no more values, so backtrack. Conflict set is $\{A_1, H, A_4\}$ so jump back to A_4. Add $\{A_1, H\}$ to A_4's conflict set.

j. $A_4 = G$ conflicts with H, so $A_4 = B$.

k. $F_1 = R$

l. $A_2 = R$ conflicts with A_1, $A_2 = G$ conflicts with H, so $A_2 = B$.

m. $F_2 = R$

n. $A_3 = R$.

o. $T = R$ conflicts with F_1 and F_2, $T = G$ conflicts with G, so $T = B$.

p. Success.

Exercise 5.XYAC

Consider a CSP with variables X, Y with domains $\{1, 2, 3, 4, 5, 6\}$ for X and $\{2, 4, 6\}$ for Y, and constraints $X < Y$ and $X + Y > 8$. List the values that will remain in the domain of X after enforcing arc consistency for the arc $X \to Y$ (which prunes the domain of X, not Y).

The resulting domain of X is $\{3, 4, 5\}$.

Exercise 5.CPTF

Are the following statements true or false?

a. Running forward checking after the assignment of a variable in backtracking search will ensure that every variable is arc consistent with every other variable.

b. In a CSP constraint graph, a link (edge) between any two variables implies that those two variables may not take on the same values.

c. For any particular CSP constraint graph, there exists exactly one minimal cutset.

d. Two different CSP search algorithms may give different results on the same constraint satisfaction problem.

e. A CSP can only have unary and binary constraints.

f. If forward checking eliminates an inconsistent value, enforcing arc consistency would eliminate it as well.

g. If enforcing arc consistency eliminates an inconsistent value, forward checking would eliminate it as well.

h. When enforcing arc consistency in a CSP, the set of values which remain when the algorithm terminates depends on the order in which arcs are processed from the queue.

a. False. Forward checking only checks variables that are connected to the variable being assigned.

b. False. A link represents any constraint, including, e.g., equality!

c. False. There may be several minimal cutsets of equal size. This is common when the graph has symmetries.

d. True. A CSP may have many solutions, and which one is found first depends on the order in which the space of assignments is searched. If there is only one solution, then all algorithms should agree.

e. False. There are several examples in the book and elsewhere of constraints with more than two variables, including cryptarithmetic and sudoku problems. (It is possible to reduce any n-ary constraint into a set of binary constraints.)

f. True.

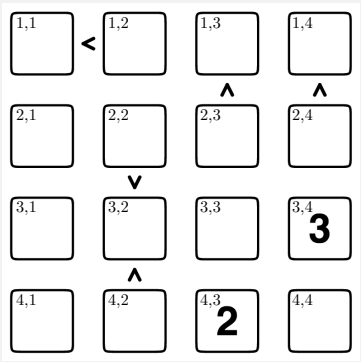
g. False.

h. False. Arc consistency eliminates values which cannot be possible, it does not make choices when there are several possibilities, so any order will end up with the same results.

Exercise 5.FUTO

Futoshiki is a Sudoku-like Japanese logic puzzle that is very simple, but can be quite challenging. You are given an n x n grid, and must place the numbers $1, \ldots n$ in the grid such that every row and column has exactly one of each. Additionally, the assignment must satisfy the inequalities placed between some adjacent squares. The inequalities apply only to the two adjacent squares, and do not directly constrain other squares in the row or column.

To the right is an instance of this problem, for size $n = 4$. Some of the squares have known values.



Let's formulate this puzzle as a CSP. We will use 4^2 variables, one for each cell, with X_{ij} as the variable for the cell in the ith row and jth column. The only unary constraints will be those assigning the known initial values to their respective squares (e.g. $X_{34} = 3$).

a. Complete the formulation of the CSP. Describe the domains of the variables, the two unary constraints, and all binary constraints you think are necessary. You can describe the constraints using concise mathematical notation. Do not use general n-ary constraints (such as *alldiff*).

b. After enforcing unary constraints, consider the binary constraints relating X_{14} and X_{24}. Enforce arc consistency on just these constraints and state the resulting domains for the two variables.

c. Suppose we enforced unary constraints and ran arc consistency on this CSP, pruning the domains of all variables as much as possible. After this, what is the maximum possible domain size for any variable? *Hint*: consider the least constrained variable(s); you should *not* have to run every step of arc consistency to answer this.

d. Suppose we enforced unary constraints and ran arc consistency on the initial CSP in the figure above. What is the maximum possible domain size for a variable adjacent to an inequality?

e. By inspection of column 2, we find it is necessary that $X_{32} = 1$, despite not having found an assignment to any of the other cells in that column. Would running arc consistency find this requirement? Explain why or why not.

a. Domains: $X_{ij} \in \{1, 2, 3, 4\}, \forall i, j$
Unary constraints: $X_{34} = 3, X_{43} = 2$
Inequality binary constraints: $X_{11} < X_{12}$, $X_{13} < X_{23}, X_{14} < X_{24}$, $X_{32} < X_{22}$, $X_{32} < X_{42}$
Row binary contraints: $X_{ij} \neq X_{ik}, \forall i, j, k, j \neq k$
Column binary contraints: $X_{ij} \neq X_{kj}, \forall i, j, k, i \neq k$

b. $X_{14} \in \{1, 2\}$, $X_{24} \in \{2, 4\}$. Note that both threes are removed from the column constraint with X_{34}.

c. The maximum possible domain size is 4 (ie, no values are removed from the original domain). Consider X_{21}. We will not be able to eliminate any values from its domain through arc consistency.

d. The maximum domain size is 3. An inequality constraint always eliminates either 1 or 4 from a variable's domain.

e. No, arc consistency would not find this requirement. Enforcing the $X_{32} \rightarrow X_{42}$ arc and the $X_{42} \rightarrow X_{43}$ arc leaves X_{42} with a domain of $\{3, 4\}$. Enforcing the $X_{32} < X_{22}$ constraints leaves $X_{32} \in \{1, 2, 3\}$ and $X_{22} \in \{2, 3, 4\}$. Enforcing that they are all different does not remove any values. After this point, every arc in this column is consistent and X_{32} is not required to be 1.

5.3 Backtracking Search for CSPs

Exercise 5.MCON

Explain why it is a good heuristic to choose the variable that is *most* constrained but the value that is *least* constraining in a CSP search.

The most constrained variable makes sense because it chooses a variable that is (all other things being equal) likely to cause a failure, and it is more efficient to fail as early as possible (thereby pruning large parts of the search space). The least constraining value heuristic makes sense because it allows the most chances for future assignments to avoid conflict.

Exercise 5.MAPR

Generate random instances of map-coloring problems as follows: scatter n points on the unit square; select a point X at random, connect X by a straight line to the nearest point Y such that X is not already connected to Y and the line crosses no other line; repeat the previous step until no more connections are possible. The points represent regions on the

map and the lines connect neighbors. Now try to find k-colorings of each map, for both $k = 3$ and $k = 4$, using min-conflicts, backtracking, backtracking with forward checking, and backtracking with MAC. Construct a table of average run times for each algorithm for values of n up to the largest you can manage. Comment on your results.

Exercise 5.ACTA

Use the AC-3 algorithm to show that arc consistency can detect the inconsistency of the partial assignment $\{WA = green, V = red\}$ for the problem shown in Figure 5.1.

We'll trace through each iteration of the **while** loop in AC-3 (for one possible ordering of the arcs):

a. Remove $SA - WA$, delete G from SA.

b. Remove $SA - V$, delete R from SA, leaving only B.

c. Remove $NT - WA$, delete G from NT.

d. Remove $NT - SA$, delete B from NT, leaving only R.

e. Remove $NSW - SA$, delete B from NSW.

f. Remove $NSW - V$, delete R from NSW, leaving only G.

g. Remove $Q - NT$, delete R from Q.

h. Remove $Q - SA$, delete B from Q.

i. remove $Q - NSW$, delete G from Q, leaving no domain for Q.

Exercise 5.ABCT

Consider a CSP with three variables: A, B, and C. Each of the three variables can take on one of two values: either 1 or 2. There are three constraints: $A \neq B$, $B \neq C$, and $A \neq C$. What values for what variables would be eliminated by enforcing *arc-consistency*? Explain your answer.

No values would be eliminated by arc-consistency. There is no solution to this CSP, but arc-consistency does not detect this. For every individual arc either value has a consistent assignment.

Exercise 5.ACWC

What is the worst-case complexity of running AC-3 on a tree-structured CSP?

On a tree-structured graph, no arc will be considered more than once, so the AC-3 algorithm is $O(ED)$, where E is the number of edges and D is the size of the largest domain.

Exercise 5.ACTF

AC-3 puts back on the queue *every* arc (X_k, X_i) whenever *any* value is deleted from the domain of X_i, even if each value of X_k is consistent with several remaining values of X_i. Suppose that, for every arc (X_k, X_i), we keep track of the number of remaining values of X_i that are consistent with each value of X_k. Explain how to update these numbers efficiently and hence show that arc consistency can be enforced in total time $O(n^2d^2)$.

The basic idea is to preprocess the constraints so that, for each value of X_i, we keep track of those variables X_k for which an arc from X_k to X_i is satisfied by that particular value of X_i. This data structure can be computed in time proportional to the size of the problem representation. Then, when a value of X_i is deleted, we reduce by 1 the count of allowable values for each (X_k, X_i) arc recorded under that value. This is very similar to the forward chaining algorithm in Chapter 7. See Mohr and Henderson (1986) for detailed proofs.

5.4 Local Search for CSPs

Exercise 5.SUDK

We introduced Sudoku as a CSP to be solved by search over partial assignments because that is the way people generally undertake solving Sudoku problems. It is also possible, to attack Sudoku problems with local search over complete assignments. How well would a local solver using the min-conflicts heuristic do on Sudoku problems?

It is certainly possible to solve Sudoku problems in this fashion. However, it is not as effective as the partial-assignment approach, and not as effective as min-conflicts is on the N-queens problem. Perhaps that is because there are two different types of conflicts: a conflict with one of the numbers that defines the initial problem is one that must be corrected, but a conflict between two numbers that were placed elsewhere in the grid can be corrected by replacing either of the two. A version of min-conflicts that recognizes the difference between these two situations might do better than the naive min-conflicts algorithm.

Exercise 5.CSLS

Suppose we are solving a CSP with local search. The problem has N variables, each of which has a domain with d values (for example, in Sudoku, $N = 81$ and $d = 9$). We are interested in the successors of any state in the local search: from a given assignment to variables, what other assignments will we consider, from which we will choose the one with the minimal conflicts. Using $O()$ notation, describe the computational complexity in terms of N and d for each of the following approaches to generating successors:

a. Randomly choose a variable, then consider all assignments to that variable.

b. For the variable with the most conflicts, consider all assignments to it.

c. Consider all states that differ by the assignment to exactly one variable.
d. Consider all states that differ by the assignment to one or two variables. (This can be useful to escape from plateaus).

a. $O(d)$
b. $O(d+N)$
c. $O(Nd)$
d. $O(N^2d^2)$

Exercise 5.RCSP
Using a CSP solver program and another program to generate random problem instances of CSPs, report on the time to solve the problem as a function of the ratio of the number of constraints to the number of variables.

Details will vary somewhat depending on exactly the structure of the random problems, but solving should be fast except fro a small critical range of the constraints-to-variables ratio.

Exercise 5.BCTA
Ali, Bo, Cleo, and Dallas are picking their entrees at a restaurant. The choices are pasta, quesadillas, risotto, and sushi. They have some strict dietary preferences:

- Cleo will not order sushi.
- Ali and Bo want to steal each other's food, so they will order different dishes.
- Bo likes carbs, so he will only order pasta or risotto.
- Cleo likes to be unique in her food orders and will not order the same dish as anybody else, with one exception: Ali and Cleo are actually twins, and always order the same dish as each other.
- Dallas really dislikes quesadillas and will not order them.

Answer the following questions for this situation:

a. If we formulate this as a CSP with variables $\{A, B, C, D\}$,e ach with domain $\{P, Q, R, S\}$, what are the constraints?
b. We will run basic backtracking search to solve this CSP and make sure that every person (variable) is matched with their dream dish (value). We will select unassigned variables in alphabetical order, and we will also iterate over values in alphabetical order. What assignment will backtracking search return?
c. Now assume that no values have been assigned to the variables and we will run one iteration of forward checking. What value(s) will be eliminated for which variables if we assign "pasta" to Ali?

d. Now assume we will solve the problem with local search using the min-conflicts algorithm. Assume we start with the initial assignment $\{A = P, B = P, C = P, D = P\}$ and choose B as the variable to change. How many conflicts does the current value of B pose? What value for B would minimize the number of conflicts?

a. $C \neq S, A \neq B, B \in \{P, R\}, C \notin \{B, D\}, D \neq Q$.
b. $\{A = P, B = R, C = P, D = R\}$.
c. Eliminate P from B; eliminate Q, S, R from C; eliminate nothing from D.
d. $B = S$ conflicts with $A \neq B, B \in \{P, R\}$, and $C \notin \{B, D\}$, so 3 conflicts. Setting $B = R$ would resolve all 3 of B's conflicts; no other value choice would do that.

5.5 The Structure of Problems

Exercise 5.TCSP

The TREE-CSP-SOLVER (Figure 5.10) makes arcs consistent starting at the leaves and working backwards towards the root. Why does it do that? What would happen if it went in the opposite direction?

We establish arc-consistency from the bottom up because we will then (after establishing consistency) solve the problem from the top down. It will always be possible to find a solution (if one exists at all) with no backtracking because of the definition of arc consistency: whatever choice we make for the value of the parent node, there will be a value for the child.

Exercise 5.CSPE

Define in your own words the terms constraint, commutativity, backtracking search, arc consistency, backjumping, min-conflicts, and cycle cutset.

A **constraint** is a restriction on the possible values of two or more variables. For example, a constraint might say that $A = a$ is not allowed in conjunction with $B = b$.

Two actions are **commutative** if they have the same effect no matter which order they are performed in.

Backtracking search is a form of depth-first search in which there is a single representation of the state that gets updated for each successor, and then must be restored when a dead end is reached.

A directed arc from variable A to variable B in a CSP is **arc consistent** if, for every value in the current domain of A, there is some consistent value of B.

Backjumping is a way of making backtracking search more efficient, by jumping back more than one level when a dead end is reached.

Min-conflicts is a heuristic for use with local search on CSP problems. The heuristic says that, when given a variable to modify, choose the value that conflicts with the fewest

number of other variables.

A **cycle cutset** is a set of variables which when removed from the constraint graph make it acyclic (i.e., a tree). When the variables of a cycle cutset are instantiated the remainder of the CSP can be solved in linear time.

Exercise 5.CGCS

Consider a CSP with a constraint graph consisting of n variables arranged in a circle, where each variable has two constraints, one with each neighbor on either side. Explain how to solve this class of CSPs efficiently, in time $O(n)$.

We can use cutset conditioning to reduce the circle to a tree structure, which can be efficiently solved without backtracking by one backward arc-enforcing pass and one forward value-setting pass. In other words, pick any variable X, set it to some value v from the domain, and execute the efficient tree-solving procedure. Repeat for all values in the domain.

Exercise 5.GCCS

Suppose that a graph is known to have a cycle cutset of no more than k nodes. Describe a simple algorithm for finding a minimal cycle cutset whose run time is not much more than $O(n^k)$ for a CSP with n variables. Search the literature for methods for finding approximately minimal cycle cutsets in time that is polynomial in the size of the cutset. Does the existence of such algorithms make the cycle cutset method practical?

A simple algorithm for finding a cutset of no more than k nodes is to enumerate all subsets of nodes of size $1, 2, \ldots, k$, and for each subset check whether the remaining nodes form a tree. This algorithm takes time $\binom{\sum_{i=1}^{k} n}{nk}$, which is $O(n^k)$.

Becker and Geiger (1994) give an algorithm called MGA (modified greedy algorithm) that finds a cutset that is no more than twice the size of the minimal cutset, using time $O(E + V \log(V))$, where E is the number of edges and V is the number of variables.

Whether the cycle cutset approach is practical depends more on the graph than on the cutset-finding algorithm. That is because, for a cutset of size c, we still have an exponential (d^c) factor before we can solve the CSP. So any graph with a large cutset will be intractable to solve by this method, even if we could find the cutset with no effort at all.

Exercise 5.TILS

Consider the problem of tiling a surface (completely and exactly covering it) with n dominoes (2×1 rectangles). The surface is an arbitrary edge-connected (i.e., adjacent along an edge, not just a corner) collection of $2n$ 1×1 squares (e.g., a checkerboard, a checkerboard with some squares missing, a 10×1 row of squares, etc.).

a. Formulate this problem precisely as a CSP where the dominoes are the variables.

b. Formulate this problem precisely as a CSP where the squares are the variables, keeping the state space as small as possible. (*Hint:* does it matter which particular domino goes

on a given pair of squares?)

c. Construct a surface consisting of 6 squares such that your CSP formulation from part (b) has a *tree-structured* constraint graph.

d. Describe exactly the set of solvable instances that have a tree-structured constraint graph.

The key aspect of any correct formulation is that a complete assignment satisfying the constraints must be a real solution to the real problem.

a. Variable domains: all pairs of adjacent squares. You might want to avoid having the same pair of squares appearing twice in different orders.
Constraints: every pair of variables is connected by a constraint stating that their values may not overlap (i.e., they cannot share any square).

b. Variable domains: the set of (up to four) adjacent squares. The idea is that the domino covering this square can choose exactly one of the adjacent squares to cover too.
Constraints: between every pair of adjacent squares A and B. A can have value B iff B has value A.

c. The constraints in (b) are binary constraints relating adjacent squares, so they will form a loop whenever the squares form a loop (e.g., any 2 by 2 block). So any problem where the "width" of the shape is 1 is OK, e.g., 6 in a row, and there are no loops.

Exercise 5.BCSP

In a general constraint satisfaction problem with N binary-valued variables, what is the minimum, and the maximum number of times that backtracking search will backtrack, expressed in $O()$ notation (i.e. $O(1), O(n^2)$, etc.).

The minimum is $O(1)$; if we get lucky there may be no backtracking at all. The maximum is $O(2^n)$; we might get unlucky and every assignment except the last one needs backtracking. There are 2^N assignments of N binary variables.

Exercise 5.SPCS

Suppose you have a state-space search problem defined by the usual stuff:

- a set of states s;
- an initial state s_0;
- a set of actions A including the $NoOp$ action that has no effect;
- a transition model $Result(s, a)$;
- a set of goal states G.

Unfortunately, you have no search algorithms! All you have is a CSP solver. How could you reformulate this as a CSP? You may assume that you are given the maximum number of

steps, T that any plan can have. Make sure that your formulation makes it easy to see what the plan is.

The straightforward solution is to have variables $S_0, \ldots, S_T$ for the state at each time step, with the domain of each being the set of states s; and variables $A_0, \ldots, A_{T-1}$ for the action to be taken at each time step, with the domain of each being A. The constraints are

- $S_0 = s_0$;
- $S_T \in G$;
- For t from 0 to $T-1$, $S_{t+1} = Result(S_t, A_t)$.

This would always find a solution of T steps if there is one; if there are shorter solutions it would pad them out with $NoOp$ actions. Alternatively, you could specify the goal as $(S_0 \in G) \vee (S_1 \in G) \vee \cdots \vee (S_T \in G)$, or you could run the CSP solver multiple times for successive values of T starting from 0.