Homework 1: Discrete Optimization & CSPs

15381: Artificial Intelligence (Fall 2013)

TAs : Subhodeep Moitra, Jesse Dunietz, Danny Zhu

OUT: Aug 27, 2013
DUE: Sep 10, 2013, 11:59 AM

1 Wedding Planning, 30 points [Jesse]

Congratulations! Your cousin is getting married, and she won't have anyone but you as the wedding planner! As usual, though, family is making things difficult. The following characters need to be seated at the family table:

- Your Uncle **B**ogwald and Aunt **M**ergatroyd
- The bride's brother **S**eptimus (also your cousin), his wife **H**epsiba, and their two-year-old daughter **P**ersephone
- Mergatroyd and Bogwald's pet capybara **C**appy (it's a long story)

You label the seats around the table 1-6 and try to find a satisfactory way to arrange all the relatives. The table is circular, so seat 1 is next to seat 6. Here is what you know:

- No two animals (human or rodent) can share a seat, and no animal can take up two seats.
- Bogwald and Mergatroyd must be seated next to each other.
- Septimus and Hepsiba must be seated next to each other.
- Hepsiba must be seated next to Persephone.
- Persephone, Septimus, and Hepsiba are all allergic to capybaras.
- Bogwald insists on honoring the capybara by putting it in Seat 1.

1. There are two ways to formulate this problem as a CSP. For each, describe the variables and their domains and give the constraints. (You can represent each person/animal by the letter bolded above.) In which formulation are the constraints simpler? Why would the more complex constraints make it difficult to run AC-3? *(Hint: use the modulo operator to express the constraints.)* **[7 points]**
2. Using the formulation in which the values are numbers, run the AC-3 algorithm for arc consistency (include the queue at each step in your answer). Give the domains each variable is left with when the algorithm terminates.**[8 points]**
3. Now switch to the other formulation. Restrict the domains of each variable to the possibilities you found in the previous part. (In other words, if you previously found that assigning Person A to Seat X was impossible, and assigning A to X is represented in the new formulation by assigning variable V to value C, remove C from the domain of V.)

Draw out the search tree for a depth-first search with backtracking. Each node in your tree should be a single variable assignment (i.e., node labels are of the form $V = v$). Use no common sense in generating this tree – show every attempt you make at assigning a variable, and reject a partial assignment only if it violates the constraints. When deciding which variable to assign next or which value to assign to it, pick the one that is earliest in alphabetical or numerical order. Stop once you've assigned Hepsiba to a seat without violating any constraints. What causes you to have to keep backtracking? **[8 points]**

4. Now consider what would happen if you ran the same search with forward checking. How would this prevent excessive backtracking? **[2 points]**

5. We discussed in lecture that the order in which you choose variables to assign can have a huge impact on performance. Draw out the search tree as in part 3 (still without forward checking), but this time select the next variable using the "Minimum Remaining Values" heuristic. Try values in alphabetical order, as before. What do you discover about this constraint problem? Why was the heuristic able to expose this result so much more quickly? **[5 points]**

2 It's All Binary, 30 points [Subhodeep]

Consider a CSP with variables $X_1, X_2, \cdots, X_n$. Assume that the domain of X_i is finite.

1. Show how unary constraints of the form $X_i = c$ or $X_i \neq c$ can be eliminated by altering the domains of variables. **[5 points]**

2. Show how a ternary constraint such as $X_1 + X_2 = X_3$ can be converted into three binary constraints by using an auxiliary variable. *(Hint: Consider a new variable that takes on values that are pairs of other values, and consider constraints such as "Y is the first element of the pair Z".)* **[10 points]**

3. Generalize this result and show that constraints of the form $X_1 + X_2 + \cdots + X_{n-1} = X_n$ can also be written in terms of binary constraints. What is the minimum number of binary constraints needed in this case? **[10 points]**

4. Using the above results, briefly propose a scheme by which any higher order CSP can be converted into a binary constraint CSP. Why might it be advantageous to do this? **[5 points]**

3 Weighted N-Queens, 40 points [Jesse]

In the Weighted N-Queens problem, each square (x, y) of an N-by-N chess board is assigned some weight $w(x, y)$. We want to place N chess queens on the board such that we minimize the number of attacks (two queens sharing the same row, column, or diagonal), and the sum of weights of the squares used by the queens.

More formally, for an assignment of N queens at positions $\{(x_i, y_i)\}_{i=1}^{N}$, we wish to minimize

$$L = \sum_{i=1}^{N} w(x_i, y_i) + \sum_{i=1}^{N} \sum_{j=i+1}^{N} I(i, j) \tag{1}$$

where

$$I(i,j) = \begin{cases} 1 & \text{Queens } i \text{ and } j \text{ are attacking each other} \\ 0 & \text{Otherwise} \end{cases}$$

Write a program to minimize the loss function L. You may use any optimization method. If you use simulated annealing, note that your results will vary depending on the initial temperature T and the rate α by which it decreases, so make sure to try different combinations and see the effect on the results.

Your program should be called with the following format:

`WeightedQueens -N [num_queens] -W [weight_function_id]`

where the available weight functions are:

1. $w(x, y) = |x - y|/4$
2. $w(x, y) = \begin{cases} 0.5 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$
3. $w(x, y) = x * y/n^2$

Note that `WeightedQueens` may be a script running your actual executable (in case you prefer to use a suffix for the executable filename, e.g., `WeightedQueens.jar`), but make sure the above commands are running as is, since we may be auto-grading your code.

Your output should include the parameters of the experiment (N and W, and T and α if applicable), the cost of the proposed assignment L, and the (x, y) assignment of each of the queens, each on a separate line. The output is expected to look exactly like this (assuming simulated annealing):

```
N=8
W=1
L=28
T=15.3
alpha=0.2
(0,0)
(1,1)
(2,2)
(3,3)
(4,4)
(5,5)
(6,6)
(7,7)
```

If you do not use simulated annealing, you can simply omit the T and alpha lines.

Please submit all of the following files:

1. Your source code. (TAs can handle Python, C/C++, Scheme, and Java. If you *need* to use some esoteric language, contact the TAs individually.)
2. Makefile – compiles your code if needed; otherwise does nothing.
3. README – a text file which explains the details of the algorithm that you have implemented. The first line of this file should contain the Andrew ids of both partners of this assignment, separated by a single comma.
4. `results.txt` – a text file which contains the concatenated output of your algorithm for all combinations of $N \in \{4, 6, 8, 12\}$ and $W \in \{1, 2, 3\}$. Do not add any characters between the concatenated results, as this may be auto-graded.

If your code does not completely meet the specification, partial credit will be assigned according to the following rubric:

- 20 points for the algorithm for a single iteration of local search being correct
- 15 additional points for the algorithm running correctly for a single set of parameters
- 5 additional points for the algorithm being able to handle multiple values for the above parameter (and for finding reasonable T and α values, if applicable)
- -3 points for every coding error we have to correct to make the algorithm run (e.g., makefile issues and uncaught exceptions)

3.1 Extra Credit [3 points]

We will run a competition to see whose program can solve the largest N-Queens board in 30 minutes of computation time. See how well you can do!

Submission Procedure

Submission will be electronic via AFS. You'll create a directory named `hw1` under `/afs/cs/academic/class/15381-f13/homeworks/[your andrew id]` and place all the files there.

- *Theoretical Problems:* Your answers to the written part should be submitted as a PDF file. You can either typeset your submission or scan in handwritten solutions.
- *Programming Problem:* You are encouraged to do the programming part in pairs. If you do, please only submit one copy of the required files. In your submission directory place a file named `README`. The first line of this file should contain the andrew ids of both partners separated by a single comma and nothing more (for example: jdunietz,dannyz).