CS221 Fall 2015 Homework [6]

SUNet ID: [06039473]

Name: [Lingjie Kong]

Collaborators: [list all the people you worked with]

By turning in this assignment, I agree by the Stanford honor code and declare that all of this is my own work.

**Problem 0: Warmup**

1. **Specify the variables with their domain and the constrains with tier scope and expression**

**Variable**: The variable is each switch which is represented as

**Variable domain**: its domain of a switch is a list of lights which will be controller by this switch.

**Constrains**: the constrain is that we want the total number of a specific light controller by multiple switches has a total count of toggle to be an odd number for all lights.

To turn all lights on, it must satisfy all n constrain that

1. **Let’s consider a simple SCP with 3 variables and 2 binary factors**
2. **How many consistent assignments are there for this assignment**

There are two consistent assignments with weight of 1 and there are

1. **To see why variable ordering is important by solving CSP without using any heuristics and a fixed ordering of**

As a result, without heuristic, the backtrack will be called for 9 times

{}

1. **To see why lookahead can be used by using AC-3**

As a result, with heuristic AC-3, the backtrack will be called for 7 times

{}

1. **Implement create\_chain\_csp() by creating a generic chain CSP and XOR as factors.**

See code in submission.py

**Problem 1: CSP solving**

1. See code in submission.py
2. See code in submission.py
3. See code in submission.py

**Problem2: Handling n-ary factors**

1. **Reduce this CSP to one with only unary and binary constrains**

Let the input be n variables X0, X1, ..., Xn.

where each "--\*--" is a binary constraint and "--^" and "--^^" are unary  
After adding auxiliary variables, the factor graph will look like this:  
^--A1--\*--A2--\*--A3--result--^

| | |  
 \* \* \*  
 | | |  
 X1 X2 X3

One need to introduce auxiliary variable A1, A2, and A3 help reduce the current ternary constrain for X1, X2, and X3 to unary and binary as shown above in the graph. Ai is (pre, current, post) pair from processing Xi

Variables:

Factors:

As shown above, as a result all the factors are unary and binary constrains.

1. See code in submission.py

**Problem 3: Course Scheduling**

1. See code in submission.py
2. See code in submission.py
3. My schedule is as below which satisfy my requirement

Units: 1-15

Quarter: ['Win2017', 'Spr2017']

Taken: set(['CS103', 'CS109', 'CS106A', 'CS107', 'CS106B', 'STATS116', 'CS106X'])

Requests:

Request{['CS149'] ['Win2017'] [] 1}

Request{['CS143'] ['Spr2017'] [] 1}

Request{['CS161'] ['Spr2017'] [] 1}

Found 22 optimal assignments with weight 1.000000 in 125 operations

First assignment took 21 operations

1.0

('sum', 'sum-up-in-Win2017', ('CS161', 'Win2017')) = (4, 0, 4)

(Request{['CS161'] ['Spr2017'] [] 1}, 'Spr2017') = CS161

('CS149', 'Spr2017') = 0

('sum', 'sum-up-in-Spr2017', ('CS143', 'Spr2017')) = (0, 0, 0)

('CS161', 'Spr2017') = 5

('sum', 'sum-up-in-Win2017', 'aggregated') = 4

('sum', 'sum-up-in-Win2017', ('CS143', 'Win2017')) = (4, 0, 4)

(Request{['CS149'] ['Win2017'] [] 1}, 'Spr2017') = None

('CS161', 'Win2017') = 0

('sum', 'sum-up-in-Spr2017', 'aggregated') = 5

('sum', 'sum-up-in-Win2017', ('CS149', 'Win2017')) = (0, 4, 4)

('sum', 'sum-up-in-Spr2017', ('CS161', 'Spr2017')) = (0, 5, 5)

('sum', 'sum-up-in-Spr2017', ('CS149', 'Spr2017')) = (0, 0, 0)

('CS143', 'Win2017') = 0

('CS143', 'Spr2017') = 0

(Request{['CS143'] ['Spr2017'] [] 1}, 'Win2017') = None

(Request{['CS149'] ['Win2017'] [] 1}, 'Win2017') = CS149

(Request{['CS143'] ['Spr2017'] [] 1}, 'Spr2017') = None

('CS149', 'Win2017') = 4

(Request{['CS161'] ['Spr2017'] [] 1}, 'Win2017') = None

Here's the best schedule:

Quarter Units Course

Win2017 4 CS149

Spr2017 5 CS161