

CS221 Fall 2018 Homework [scheduling]

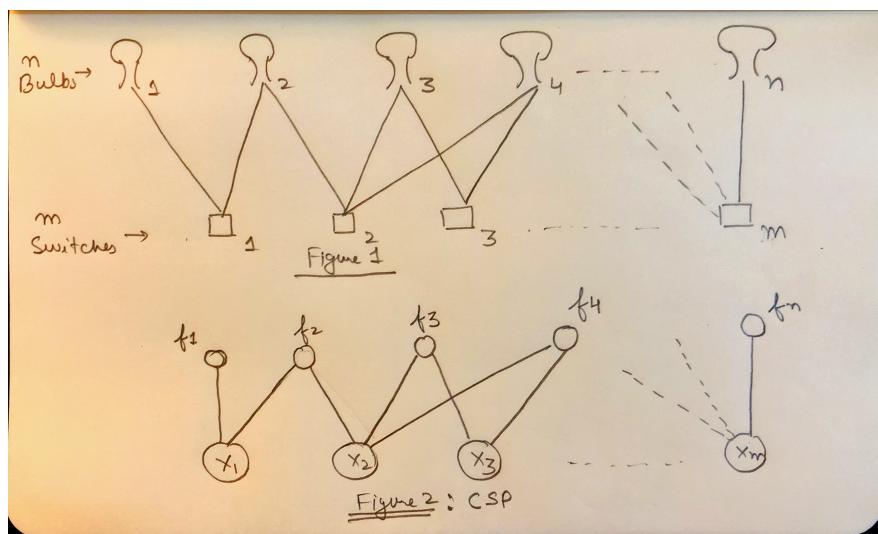
SUNet ID: prabhjot

Name: Prabhjot Singh Rai

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

Problem 0

- (a) The problem statement can be visualised as below:



Variables: The variables for the CSP are the m switches, X_1, X_2, \dots, X_m . The domain of these variables are Domains $\epsilon\{0, 1\}$, 0 being the "off" state for a switch and 1 being the "on" state.

Constraints: The constraints or the factors are created among a bulb and it's controlling switch(es). For example, from above Figure 1, if switch 1 controls bulb 1 and bulb 2, and switch 2 controls bulb 2 and bulb 3, a factor corresponding to bulb 1 would have switch 1 value as it's parameter (be dependent on switch 1) and bulb 2 would have values of both switch 1 and switch 2 as it's parameters. Since T_j (for each button $j = 1, \dots, m$) defines every set of light bulbs a switch controls, factor f_k (for each light bulb $k = 1, \dots, n$) would depend on variable j if k in T_j . The value of this factor should return an odd number, so that even numbers render the state of the bulb to be "off" and last number makes the bulb "on".

$$f_k(X_1[k \text{ in } T_1], X_2[k \text{ in } T_2], \dots, X_m[k \text{ in } T_m]) \\ = \text{sum}(X_1[k \text{ in } T_1], X_2[k \text{ in } T_2], \dots, X_m[k \text{ in } T_m]) \% 2 == 1$$

Table 1: Table depicting consistency calculation for XOR operation

x1	x2	x3	t1(x)	t2(x)	Consistency
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	1	1
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	0	1	0
1	1	1	0	0	0

- (b)
 - i. For finding consistent assignments, we draw the table for finding values of t_1 and t_2 . From the Table 1, we can see that there are two consistent solutions for x_1, x_2, x_3 , one being $\{0, 1, 0\}$ and other being $\{1, 0, 1\}$, respectively.
 - ii. For fixed variables X_1, X_2, X_3 , backtrack will be called in following ways:

```

backtrack( $\phi, 1, \{x_1 : [0, 1], x_2 : [0, 1], x_3 : [0, 1]\}$ )
backtrack( $\{x_1 : 0\}, 1, \{x_1 : [0, 1], x_2 : [0, 1], x_3 : [0, 1]\}$ )
backtrack( $\{x_1 : 0, x_3 : 0\}, 1, \{x_1 : [0, 1], x_2 : [0, 1], x_3 : [0, 1]\}$ )
backtrack( $\{x_1 : 0, x_2 : 1, x_3 : 0\}, 1, \{x_1 : [0, 1], x_2 : [0, 1], x_3 : [0, 1]\}$ )
backtrack( $\{x_1 : 0, x_3 : 1\}, 1, \{x_1 : [0, 1], x_2 : [0, 1], x_3 : [0, 1]\}$ )
backtrack( $\{x_1 : 1\}, 1, \{x_1 : [0, 1], x_2 : [0, 1], x_3 : [0, 1]\}$ )
backtrack( $\{x_1 : 1, x_3 : 0\}, 1, \{x_1 : [0, 1], x_2 : [0, 1], x_3 : [0, 1]\}$ )
backtrack( $\{x_1 : 1, x_3 : 1\}, 1, \{x_1 : [0, 1], x_2 : [0, 1], x_3 : [0, 1]\}$ )
backtrack( $\{x_1 : 1, x_2 : 0, x_3 : 1\}, 1, \{x_1 : [0, 1], x_2 : [0, 1], x_3 : [0, 1]\}$ )

```

Therefore, backtrack algorithm is called 9 times.

- iii. When lookahead is enabled (AC3):

```

backtrack( $\phi, 1, \{x_1 : [0, 1], x_2 : [0, 1], x_3 : [0, 1]\}$ )
backtrack( $\{x_1 : 0\}, 1, \{x_1 : [0], x_2 : [1], x_3 : [0]\}$ )
backtrack( $\{x_1 : 0, x_3 : 0\}, 1, \{x_1 : [0], x_2 : [1], x_3 : [0]\}$ )
backtrack( $\{x_1 : 0, x_2 : 1, x_3 : 0\}, 1, \{x_1 : [0], x_2 : [1], x_3 : [0]\}$ )
backtrack( $\{x_1 : 1\}, 1, \{x_1 : [1], x_2 : [0], x_3 : [1]\}$ )
backtrack( $\{x_1 : 1, x_3 : 1\}, 1, \{x_1 : [1], x_2 : [0], x_3 : [1]\}$ )
backtrack( $\{x_1 : 1, x_2 : 0, x_3 : 1\}, 1, \{x_1 : [1], x_2 : [0], x_3 : [1]\}$ )

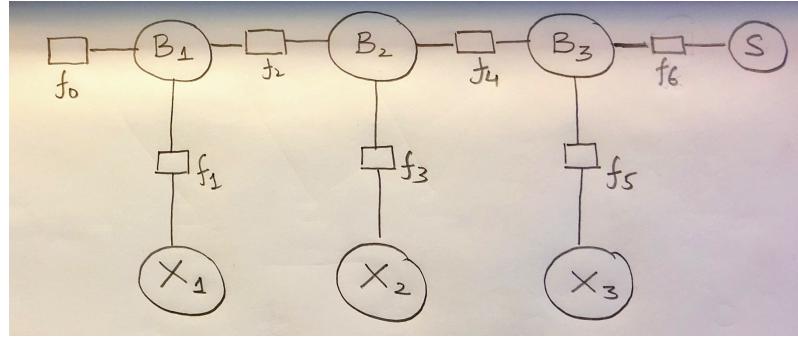
```

Therefore, backtrack algorithm with AC3 is called 7 times.

Problem 2

- (a) **Reducing the CSP:** In order to reduce an N-ary CSP to unary or binary factors, we can think of introducing another set of variables (B_1, B_2, B_3 and S) called auxiliary variables to keep a track of total sum of variables X_1, X_2 and X_3 as we traverse in the graph.

Auxiliary variables introduction: Following is the graphical representation of auxiliary variables B_1, B_2, B_3 and S and given variables X_1, X_2 and X_3 .



Domains: X_1, X_2 and X_3 are the given variables with domains $\{0, 1, 2\}$ each. As discussed earlier, auxiliary variables B_1, B_2 and B_3 keep a track of the total sum. Therefore, these are two dimensional, one element carrying the total sum of previous auxiliary variable and other keeping a record after adding value of attached variable X_i . Both $B_n[0]$ and $B_n[1]$ will have a range from $0, 1, \dots, K$, where K being the maximum sum constraint. S is the sum variable, having domain $0, 1, \dots, K$ since it tries to bind the variables based on given total sum.

Constraints: There are going to be unary and binary constraints as follows(referencing above image):

f_0 states that $B_1[0]$ should be 0, since we start with sum = 0. Therefore, $f_0(B_1) = B_1[0] == 0$

f_1, f_3 and f_5 make sure that adding X_i to first coordinate of B_i gives us second coordinate of B_i . Hence, $f_1(B_1, X_1) = B_1[0] + X_1 == B_1[1]$, similarly for $f_3(B_2, X_2) = B_2[0] + X_2 == B_2[1]$, and for $f_5(B_3, X_3) = B_3[0] + X_3 == B_3[1]$.

f_2 and f_4 make sure that first index of B_{i+1} is equal to last index of B_i . Hence, $f_2(B_2, B_1) = B_2[0] == B_1[1]$ and $f_4(B_3, B_2) = B_3[0] == B_2[1]$.

f_6 maintains that last index of B_3 and the value of sum variable S are equal. Therefore, $f_6(B_3, S) = B_3[1] == S$.

This scheme works because it has added maximum value constraint over the sum of all three variables through auxiliary sum variable S , along with propagating sum of variables one by one from X_1 to X_3 using B_1 , B_2 and B_3 .

Problem 3

- (c) Since I am planning to pursue graduate certificate by taking up courses every quarter(except summer break), with propensity towards CS221 and CS224N, I listed out all the courses offered and let the system suggest what courses should I pick up when. Here's the profile I created:

```
# Unit limit per quarter. You can ignore this for the first
# few questions in problem 2.
minUnits 3
maxUnits 3

# These are the quarters that I need to fill. It is assumed that
# the quarters are sorted in chronological order.
register Aut2018
register Win2019
register Spr2019
register Aut2019

# Courses I've already taken
taken CS107
taken CS103
taken CS106X
taken CS106B
taken CS103
taken CS124

# Courses that I'm requesting
request CS221 in Aut2018 weight 2
request CS157
request CS223A
request CS224N in Aut2019 weight 2
request CS228
request CS229
request CS231A
request CS227B
```

Table 2: Table Depicting best schedule suggested by system

Quarter	Units	Course
Aut2018	3	CS221
Win2019	3	CS223A
Spr2019	3	CS227B
Aut2019	3	CS224N

The best schedule that the system suggested is depicted in Table 2. It makes sense because I had given higher weights to CS221 and CS224N, since I personally wanted to take these courses up with a vision to finish the graduate certificate as soon as possible. As I am working, I wanted to restricted the courses to 1 per quarter, therefore added maxUnits and minUnits for this profile as 3.