CS221 Autumn 2016 Homework: Course Scheduling

SUNet ID: [06033811]
Name: [Chao Xu]

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

Problem 0: Warm-up

(a) Let's create a CSP. Suppose you have n light bulbs, where each light bulb $i = 1, \cdots, n$ is initially off. You also have m buttons which control the lights. For each button $j = 1, \cdots, m$, we know the subset $T_j \subseteq 1, \cdots, n$ of light bulbs that it controls. When button j is pressed, it toggles the state of each light bulb in T_j (For example, if $3 \subseteq T_j$ and light bulb 3 is off, then after the button is pressed, light bulb 3 will be on, and vice versa). Your goal is to turn on all the light bulbs by pressing a subset of the buttons. Construct a CSP to solve this problem. Your CSP should have m variables and n constraints. For this problem only, you can use n-ary constraints. Describe your CSP precisely and concisely. You need to specify the variables with their domain, and the constraints with their scope and expression. Make sure to include T_j in your answer.

The variables for this problem would be these m button, as X_j, where $j = 1, \cdots, m$. The domain for each variable X_j is the subset $T_j \subseteq 1, \cdots, n$ of light bulbs that it controls. The constraints are from n light bulbs T_j with each has a scope of set of buttons it depends on. In details, the first constraint is that when button j is pressed, it toggles the state of each light bulb:

$$f(T_j) = -f(T_j) \text{ if button j is pressed}$$

The second constrain is that all the light has to been 1 or at on state:

$$\forall f(T_j) = 1$$

(b) Part B

i. How many consistent assignments are there for this CSP?

There are two consistent assignment as following:

$$\{0, 1, 0\} \text{ and } \{1, 0, 1\}$$

ii. To see why variable ordering is important, let's use backtracking search to solve the CSP without using any heuristics (MCV, LCV, AC-3) or lookahead. How many times will backtrack() be called to get all consistent assignments if we use the fixed ordering X_1, X_3, X_2? Draw the call stack for backtrack(). (You should use the Backtrack algorithm from the slides. The initial arguments are $x = \emptyset$, w=1, and the original Domain.)

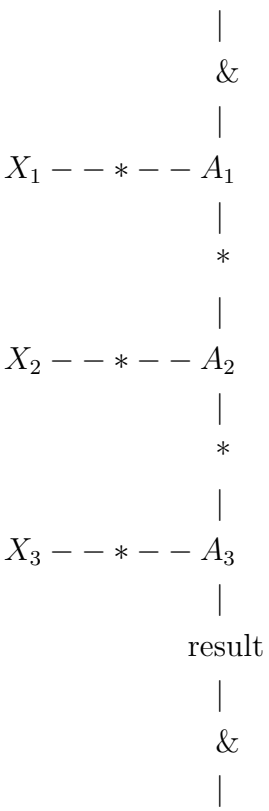
Call stack for backtrack():
Backtrack()
Backtrack($X_1 : 0$)
Backtrack($X_1 : 0$ and $X_3 : 0$)
Backtrack($X_1 : 0$ and $X_3 : 0$ and $X_2 : 1$)
Backtrack($X_1 : 0$ and $X_3 : 1$)
Backtrack($X_1 : 1$)
Backtrack($X_1 : 1$ and $X_3 : 0$)
Backtrack($X_1 : 1$ and $X_3 : 1$)
Backtrack($X_1 : 1$ and $X_3 : 1$ and $X_2 : 0$)
The backtrack function is called nine times.

iii. To see why lookahead can be useful, let's do it again with the ordering X_1, X_3, X_2 and AC-3. How many times will Backtrack be called to get all consistent assignments? Draw the call stack for backtrack(). Backtrack()
Backtrack($X_1 : 0$)
Backtrack($X_1 : 0$ and $X_3 : 0$)
Backtrack($X_1 : 0$ and $X_3 : 0$ and $X_2 : 1$)
Backtrack($X_1 : 1$)
Backtrack($X_1 : 1$ and $X_3 : 1$)
Backtrack($X_1 : 1$ and $X_3 : 1$ and $X_2 : 0$)
The backtrack function with AC-3 is called seven times, where Backtrack($X_1 : 0$ and $X_3 : 1$) and Backtrack($X_1 : 1$ and $X_3 : 0$) are ignored as there does not exist a result that can satisfied the requirement based on these assignments.

Problem 2: Handling n-ary factors

(a) Suppose we have a CSP with three variables X_1, X_2, X_3 with the same domain $\{0, 1, 2\}$ and a ternary constraint [X1+X2+X3 $\leq$ K]. How can we reduce this CSP to one with only unary and/or binary constraints? Explain what auxiliary variables we need to introduce, what their domains are, what unary/binary factors you'll add, and why your scheme works. Add a graph if you think that'll better explain your scheme.

We need to introduce the following auxiliary variables:
A_1, A_2, A_3, which consist past, current and post pair of X_is, to reduce the ternary constraint [X1+X2+X3 $\leq$ K] to one with only unary and binary constraints. The following is the factor graph with '–*–' representing binary constraint and $'--\&--'$ representing unary constraint:



The general factors are as following:
$A_1[0] == 0$
$A_i[1] == Xi$

$A_{i-1}[2] == A_i[0]$
$A_{end}[2] == result$
$result \leq K$ The domain for As are the product of all connected variables' domain above and to the left, i.e. $\{0, 0, 0\}, \ldots, \{2, 2, 4\}$ and the domain for results is from 0 to 15.

(b) See code

Problem 3: Course Scheduling

(a) See Code

(b) See Code

(c) Now try to use the course scheduler for the winter and spring (and next year if applicable). Create your own profile.txt and then run the course scheduler:
$python run p3.py profile.txt$
You might want to turn on the appropriate heuristic flags to speed up the computation. Does it produce a reasonable course schedule? Please include your $profile.txt$ and the best schedule in your writeup; we're curious how it worked out for you!

Here is my profile.txt:
Unit limit per quarter. You can ignore this for the first few questions in problem 2.
minUnits 8
maxUnits 10

These are the quarters that I need to fill out.
It is assumed that the quarters are sorted in chronological order.
register Win2017
register Spr2017

Courses I've already taken
taken ME218A
taken ME218B
taken ME218C
taken CS221

taken EE263
taken ME320
taken CS225A
taken ME203
taken ME110
taken ENGR105

Courses that I'm requesting
request ME318 in Win2017,Spr2017
request CS229 in Win2017,Spr2017
request ME300B or EE364A in Win2017
request MS&E261 in Win2017,Spr2017
request ME298 in Win2017
request CS107 in Win2017,Spr2017

And the best schedule from the profile above is as following:
Here's the best schedule:

Quarter	Units	Course
Win2017	3	EE364A
Win2017	3	MS&E261
Win2017	4	ME298
Spr2017	4	ME318
Spr2017	5	CS107