HY387 – Recitation 3

ΓΙΑΚΟΥΜΆΚΗΣ ΘΕΌΔΩΡΟΣ-ΜΆΡΙΟΣ

16/10/2020

Constraint Satisfaction Problems (CSP)

A constraint-satisfaction problem P is defined as a tuple (X,D,C), where:

X is a finite set of variables

D is a set of initial domains of the variables

C is a set of constraints over subsets of X

A <u>solution</u> to a CSP is a complete assignment to all variables that satisfies the constraints

Constraint Satisfaction Problems (CSP)

The **representational** part of a CSP is figuring out how to map a problem into variables, domains (with values), and constraints

We visualize the CSP as a **constraint graph**

- nodes representing the variables
- and the links the constraints.

The key goal is to **reduce the branching factor** to cut down search by eliminating combinations of variable values that can never participate in a solution. The whole idea is to **push for early failure** (ie, higher in the tree).

Take advantage of the problem structure better than plain search to eliminate a large fraction of the search pace, in the best case.
(In the very best case: no search remains at all – just singleton values for each variable. If multiple values – still search left to do.

Constraint Satisfaction Problems (CSP)

The goal test

- **decomposed into a set of constraints on variables**

Allows searching incrementally, and solve part of the problem first before solving the rest.

3 main techniques each using an increasing amount of ‘propagation’ of constraint.

The key questions one asks to get different methods here are

- which variable should be assigned next and in what order should the values be tried?
- what are the implications of the current var assignments for other, unassigned variables?
- when a path fails, can the search avoid repeating this failure?

Constraint Satisfaction Problems (CSP)

two sorts of methods:

- arc-consistency, which only reduces the # of values that could be in a solution
- and the other two, backtracking and forward-checking with backtracking, that can actually find solutions.

So, arc-consistency, or “Waltz labeling” really is meant to reduce search. We still must do some search in the end

Constraint Satisfaction Problems (CSP)

- **Backtracking (BT).** No constraint checks are propagated ["Assignments only"]. One variable assigned a single value at a time, and then the next variable, etc.
- **Forward-checking with backtracking (BT-FC).** A variable is assigned a unique value, and then its neighbors checked to see if their values are consistent with this assignment. [" Check neighbors only"].
- **Arc-consistency** (AKA "Waltz labeling"). Constraints are propagated through all reduced domains of variable values, until closure. This gets rid of more inconsistencies than FC, but often doesn't settle on single assignments, because n arc from X to Y in the constraint graph is consistent if, for every value of X, there is some value of Y that is consistent with X.– it could be a different y for each choice of x.

Algorithm Variants

Constraint satisfaction algorithms:

- **DFS with Backtracking (BT).** No constraint checks are propagated [“Assignments only”]. One variable is assigned a single value at a time, and then the next variable, and the next, etc. Check to see if the current partial solution violates any constraint. Whenever this check produces a zero-value domain, backup occurs.
- **DFS with Backtracking + Forward-checking**. A variable is assigned a unique value, and then only its neighbors are checked to see if their values are consistent with this assignment. That is: assume the current partial assignment, apply constraints and reduce domain of neighboring variables. [“ Check neighbors of current variable only”].
- **DFS with Backtracking + Forward-checking + propagation through singleton domains**. If during forward checking you reduce a domain to size 1 (singleton domain), then assume the assignment of the singleton domain and repeat forward checking from this variable. (Note that reduction to a singleton domain and repeated forward checking might lead to yet another singleton domain in some cases.)
- **DFS with Backtracking + Forward checking + propagation through reduced domains** (this is also known as: Arc-consistency, AC-2, since it ensures that all pairs of variables have consistent values). Constraints are propagated through all reduced domains of variable values, possibly beyond immediate neighbors of current variable, until closure. [“Propagate through reduced domains”; also called “Waltz labeling”]

Convert CSPs to binary CSPs

It is possible to convert non-binary CSPs to binary CSP.

Method 1: Dual encoding

Method 2: Hidden variable encoding

Dual Encoding

1. Each constraint *c* is represented by a *dual variable* v_c.
2. The domain of each dual variable v_c consists of the set of allowed tuples in the original constraint c.
3. Binary constraints between two dual variables exist if and only if the corresponding constraints share common variables of the original CSP.

Example

Domain: {0,1}

Constraints:

c1: $x1 + x2 + x6 = 1$

c2: $x1 – x3 + x4 = 1$

c3: $x4 + x5 - x6 \geq 1$

c4: $x2 + x5 - x6 = 0$

Example

Constraints:

c_1: $x_1 + x_2 + x_6 = 1$

c_2: $x_1 – x_3 + x_4 = 1$

c_3: $x_4 + x_5 – x_6 \geq 1$

c_4: $x_2 + x_5 – x_6 = 0$

Dual Variables:

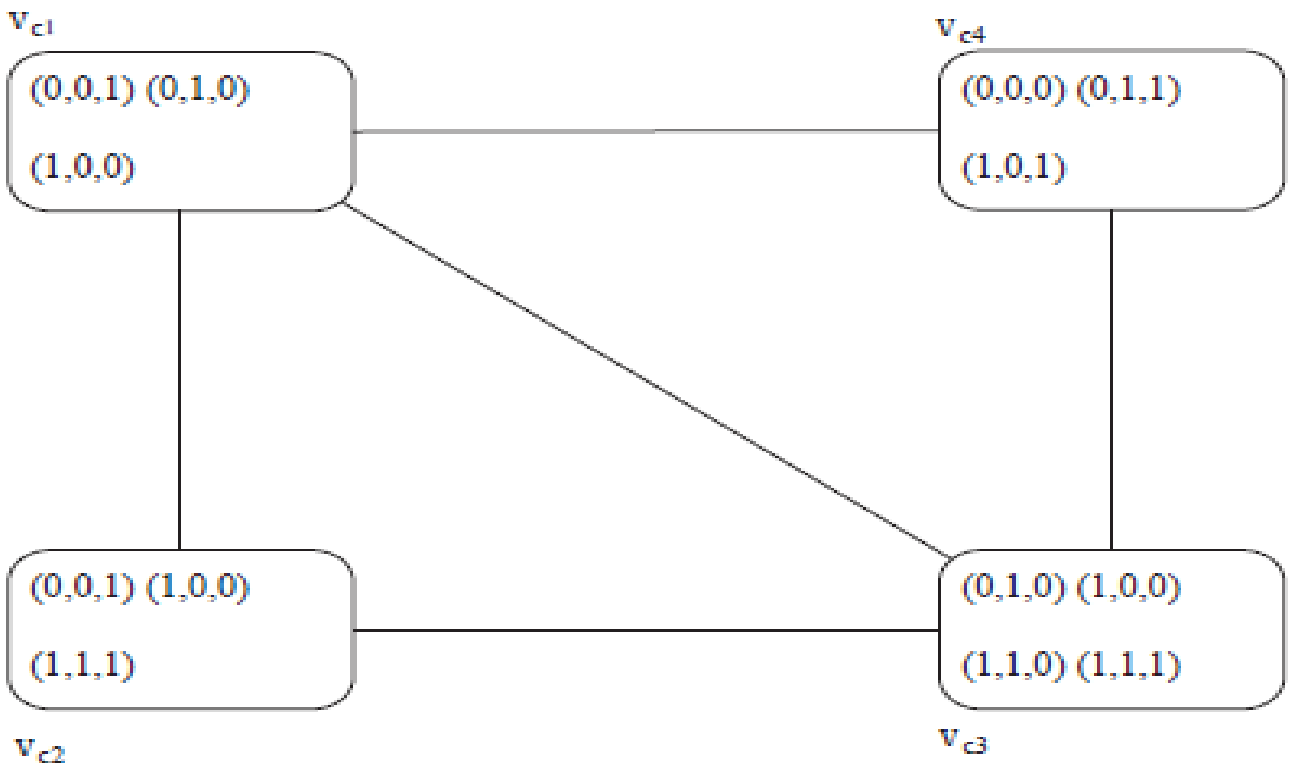
v_{c1}, domain {(0,0,1), (0,1,0), (1,0,0)}

v_{c2}, domain {(0,0,1), (1,1,1), (1,0,0)}

v_{c3}, domain {(0,1,0), (1,0,0), (1,1,0), (1,1,1)}

v_{c4}, domain {(0,0,0), (0,1,1), (1,0,1)}

Example

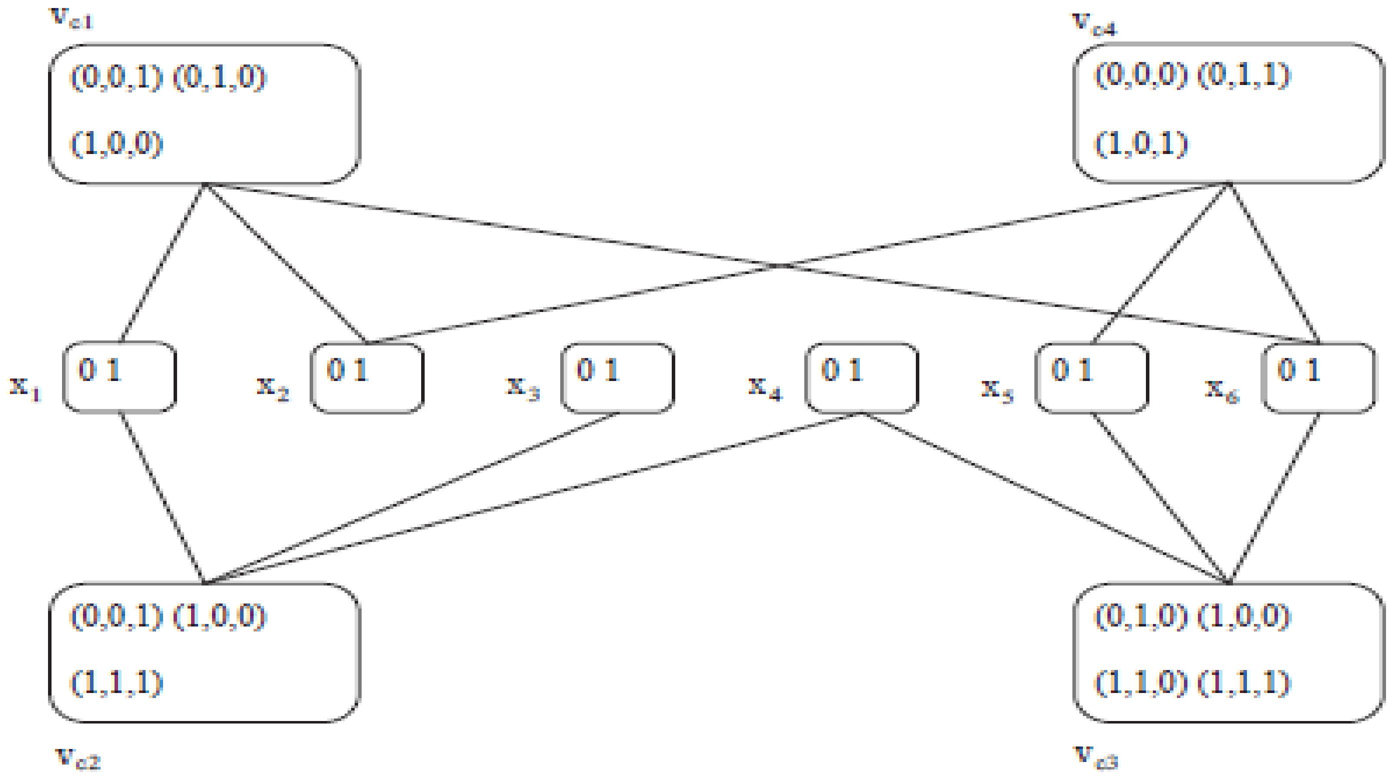


Dual encoding of the problem

Hidden Variable Encoding

1. Each constraint *c* is represented by a *dual variable* v_c. Additionally, all original variables are included.
2. The domain of each dual variable v_c consists of the set of allowed tuples in the original constraint c.
3. For each dual variable, there is a binary constraint between the dual variable and each of the original variables.

Example



Dual encoding of the problem

Boolean Satisfiability (SAT)

Problem Definition: We are given a boolean formula over variables X in CNF. Is there an assignment of values for X that satisfies the given formula?

Variables: $x_{1, \ldots,} x_n$

Domain: {0,1}

Constraints:

$(x_i \vee -x_j \vee \ldots \vee x_k) \wedge \ldots \wedge (-x_m \vee \ldots \vee x_l)$

Exercise 5.11

Show how a single ternary constraint such as “A + B = C” can be turned into binary constraints. You may assume finite domains.

Exercise 5.11

Show how a single ternary constraint such as "A + B = C" can be turned into binary constraints. You may assume finite domains.

1. Introduce an auxiliary variable AB.
2. If A and B can take values in [1,N], AB can take values in [1,N]x[1,N].
3. Use three binary constraints:
    a) A equals the first value of AB
    b) B equals the second value of AB
    c) C equals the sum of the pair of numbers of AB