

Introduction to constraint programming

Lecture II

Per Kreuger piak@sics.se &
Markus Bohlin bola@sics.se

November 2002

Constraint Satisfaction

- Constraint satisfaction problems (CSP) were first studied as a general modeling technique in AI
- Many practical combinatorial problems are straightforwardly modelled as CSPs
- A variety of techniques have been developed to solve different classes of CSPs
- From the field of CSP arose a some of the specialized techniques used within Constraint Programming (CP) to *so/ve* many practical problems

- The theory of CSPs still provide a theoretical base of CP

Constraint Satisfaction Problems (CSP)

A *Constraint Satisfaction Problem* (CSP) is a triple $\langle Z, D, C \rangle$ where:

- Z is a (finite) set of *variables* $\{x_1, \dots, x_n\}$
- D is an set of (variable) *domains* (set of possible values) for each variable in Z
 - Let, for a given variable $x \in Z$, D_x denotes its domain

- C is a set

$$\left\{ C_{\langle x_{1_i}, \dots, x_{n_i} \rangle} \right\}_{i \in |C|}$$

of *constraints*, each restricting the values that the variables $\{x_{1_i}, \dots, x_{n_i}\} \subseteq Z$ can simultaneously take

Types of CSPs

Domains may e.g. be

Booleans

Finite collections of integers, or symbols

Intervals of reals or rationals

Trees finite or rational

Constraints may be

unary a property of the object represented by the variable

binary a relation between two distinct objects

***n*-ary** a relation between *n* distinct objects

Labels

A *label* is a variable-value pair $\langle x, v \rangle$ representing the assignment of the value $v \in D_x$ to the variable x .

A *compound label* is a finite set $\{\langle x_1, v_1 \rangle, \dots, \langle x_n, v_n \rangle\}$ of labels representing the simultaneous assignment of the value $v_1 \in D_{x_1}$ to the variable x_1 , $v_2 \in D_{x_2}$ to x_2 and so on.

A *k-compound label* is a compound label of size k .

A *projection* is a subset of a compound label.

Tuple representation of constraints

A constraint $C_{\langle x_1, \dots, x_k \rangle}$ may be thought of as a set of “legal” k -compound labels

$$\{ \{ \langle x_1, v_1 \rangle, \dots, \langle x_k, v_k \rangle \} \mid C_{\langle x_1, \dots, x_k \rangle} \}$$

i.e., the set of all valid assignments

- This is a very general but wasteful representation since in general it is space exponential
- Many algorithms on CSPs are formulated on such a representation
- In most practical cases other representations are used

Satisfaction and satisfiability

A compound label L *satisfies* a constraint $C_{\langle x_1, \dots, x_k \rangle}$ if and only if there exists a projection $S_k \subseteq L$ that is a k -compound label such that

$$S \in C_{\langle x_1, \dots, x_k \rangle}$$

A CSP $\langle Z, D, C \rangle$ is *satisfiable* if and only if there exists a compound label assigning values to *all* the variables in Z that *satisfies all constraints* in C .

Solutions

A *solution* to a CSP is an assignment (compound label) of a value $v_x \in D_x$ from its domain to every variable $x \in Z$, such that every constraint is satisfied.

We may want to find:

- just one solution, with no preference as to which one
- all solutions
- the most general solution
- an optimal, or at least a good solution, given some objective function defined in terms of some or all of the variables

Tightness of constraints

Tightness of a is a measure of the difficulty of satisfying a constraint or a CSP.

Let the tightness of a constraint $C_{\langle x_1, \dots, x_n \rangle}$ in a CSP $\langle Z, C, D \rangle$ be

$$\frac{|C_{\langle x_1, \dots, x_n \rangle}|}{|D_{x_1} \times \dots \times D_{x_n}|}$$

i.e., the number of valid compound labels relative to all possible ones given only the domains of the variables.

General notions of tightness

- Constraints with low tightness are referred to as *tight*
- $x = y$ is generally a *tight* constraint since in most cases

$$\frac{|\{\{\langle x, a \rangle, \langle x, b \rangle\} \mid (a \in D_x, b \in D_y) \wedge a = b\}|}{|D_x \times D_y|} \ll 1$$

- $x \neq y$ is generally a *loose* constraint since in most cases

$$\frac{|\{\{\langle x, a \rangle, \langle x, b \rangle\} \mid (a \in D_x, b \in D_y) \wedge a \neq b\}|}{|D_x \times D_y|} \gg 0$$

Tightness of a CSP

The tightness of a CSP is number of solution divided by the total number of compound labels assigning all variables in Z .

Loose problem → many solutions, possibly a strong case for satisfiability

Tight constraints → possibly strong propagation

Tight problem → difficult to find even one solution, if most constraints are also tight we may have strong case for optimization

Problems with many loose constraints may be very tight!

Summary of essential concepts

CSP $\langle Z, D, C \rangle$ Constraint Satisfaction Problem

Domain D_x of a variable x

Label assignment $\langle x, v \rangle$ of a value v to a variable x

Compound label simultaneous assignment $\{\langle x_1, v_1 \rangle, \dots, \langle x_n, v_n \rangle\}$ of a of values v_1, \dots, v_n to variables x_1, \dots, x_n

Essential concepts (cont.)

Constraint a relation on variables (or a set of compound labels)

Satisfaction of constraint $C_{\langle x_1, \dots, x_k \rangle}$ by compound label L

$$L \supseteq S \in C_{\langle x_1, \dots, x_k \rangle}$$

Satisfiability of CSP Existence of a compound label L that satisfies all constraints

Solving a CSP Finding such a label...

Binary CSPs

A binary CSP is a CSP where all constraints are either unary $C_{\langle x \rangle}$ or binary $C_{\langle x, y \rangle}$.

Many classical solution methods work only with binary problems. In order to solve general CSPs with such techniques we can transform them to binary CSPs.

Any CSP can be transformed into a binary CSP by replacing every n -ary constraint (for $n > 2$) by¹ a new variable w with a domain containing the valid compound labels of the original constraint and n new binary

¹Other binarization methods exist. See e.g. Barták

constraints between w and each of the original variables such that the original variable is constrained to a projection of the compound labels in w .

Backside of binarization

- By transforming a general CSP into a binary CSP, we *lose* structural information about the original constraints
- Such information can potentially be used to prune the search space more aggressively by using specialized techniques for important classes non-binary constraints

Constraint graphs

Any given CSP $\langle Z, D, C \rangle$ can be represented as a *constraint graph*². In a constraint graph, each arc represents a constraint. Because an arc by definition connects at most two nodes, we need to generalize graphs to allow k -ary edges.

A *hyperedge* $\varepsilon \in \mathcal{E}$ is a set of nodes.

A *hypergraph* is a tuple $\langle V, \mathcal{E} \rangle$ where V is a set of nodes and \mathcal{E} is a set of *hyperedges*.

²In fact, by a *annotated* constraint graph, where information about constraints are associated with the edges of the graph.

Constraint hyper graphs

A *constraint hypergraph* for a CSP $\langle Z, D, C \rangle$ is the hypergraph $\langle Z, \mathcal{E} \rangle$ where Z is the set of variables in the problem and \mathcal{E} is a set of hyperedges, where each hyperedge $\varepsilon_i = \{x_{1_i}, \dots, x_{k_i}\} \in \mathcal{E}$ corresponds to a constraint $C_{\langle x_{1_i}, \dots, x_{k_i} \rangle} \in C$.

Just as general CSPs can be reduced to binary CSP, constraint hypergraphs can also be reduced to *binary constraint graphs*, which are ordinary graphs with binary and unary edges only.

Example

Consider the following CSP, where

$$Z = \{x, y, z\}$$

$$D_x = \{1, 2, 3\}, D_y = \{1, 2, 3\}, D_z = \{1, 2, 3\}$$

$$C = \{x \neq y \neq z\}$$

The constraint hyper graph of the example

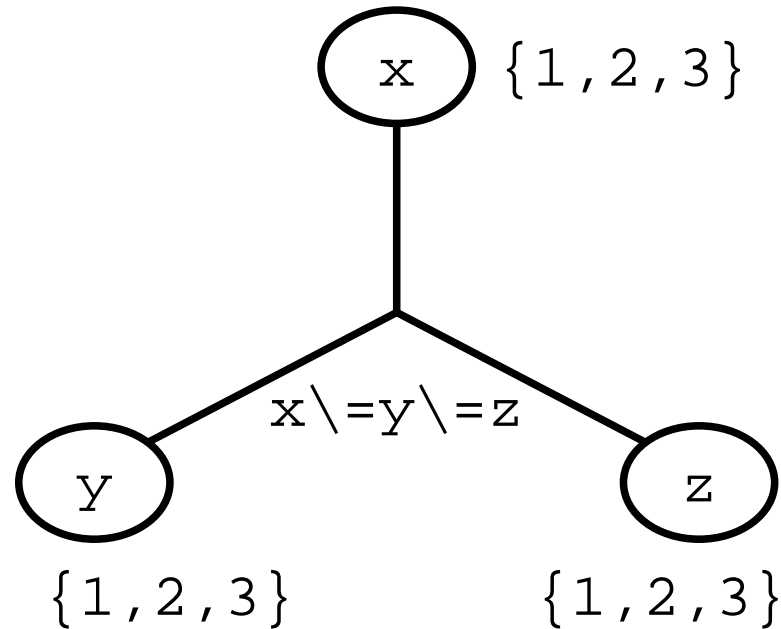


Figure 1: The constraint hyper graph $\langle \{a, b, c\}, \{\{x, y, z\}\} \rangle$

The corresponding (binary) constraint graph

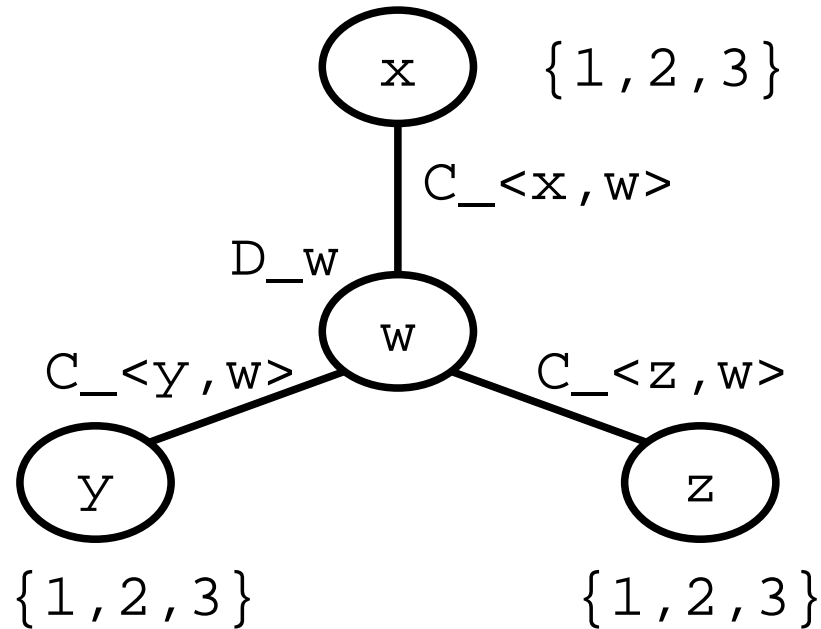


Figure 2: The constraint graph $\langle \{a, b, c, w\}, \{\{x, w\}, \{y, w\}, \{z, w\}\} \rangle$

Domain of new variable w and new constraints

where

$$D_w = \{\{\langle x, X \rangle, \langle y, Y \rangle, \langle z, Y \rangle\} \mid X \in D_x, Y \in D_y, Z \in D_z, X \neq Y \neq Z\}$$

and

$$C_{\langle x, z \rangle} = \{\{\langle x, X \rangle, \langle w, \{\langle x, X \rangle, \langle y, Y \rangle, \langle z, Z \rangle\}\}\} \mid X \in D_x, Y \in D_y, Z \in D_z\}$$

$$C_{\langle y, z \rangle} = \{\{\langle y, Y \rangle, \langle w, \{\langle x, X \rangle, \langle y, Y \rangle, \langle z, Z \rangle\}\}\} \mid X \in D_x, Y \in D_y, Z \in D_z\}$$

$$C_{\langle y,z \rangle} = \{ \{ \langle z, Z \rangle, \langle w, \{ \langle x, X \rangle, \langle y, Y \rangle, \langle z, Z \rangle \} \} \} \mid X \in D_x, Y \in D_y, Z \in D_z \}$$

A generalized graph representation

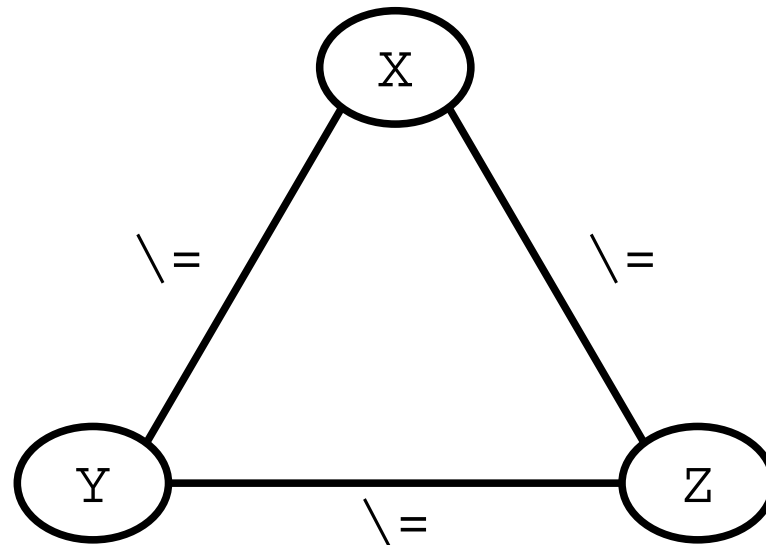


Figure 3: The constraint graph $\langle \{a, b, c, w\}, \{\{x, y\}, \{x, z\}, \{y, z\}\} \rangle$

- Many single non-binary constraint can also be represented as set of similar constraints and a given graph property, e.g. clique (Beldiceanu)

A graph coloring example

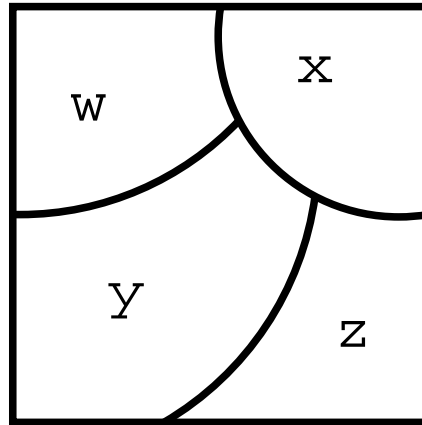


Figure 4: The map of a map coloring problem

- Three possible colors: Red, Green and Blue
- No adjacent areas in same color

Coloring graph

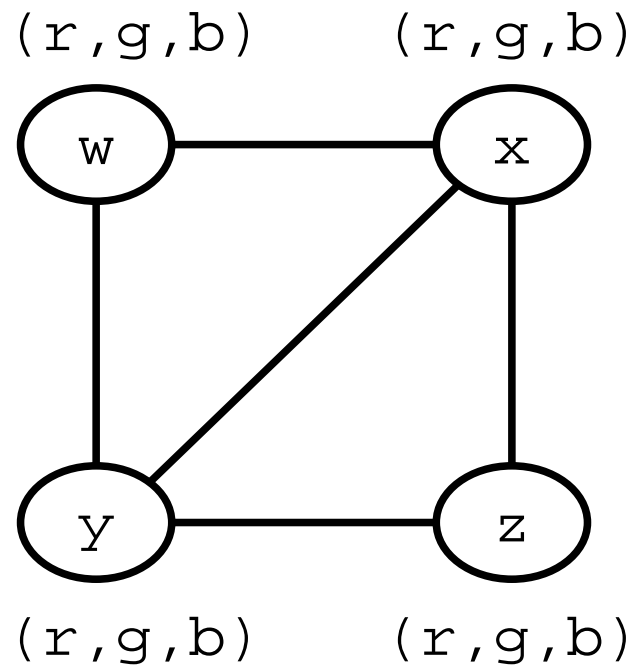


Figure 5: Map coloring graph

CSP for map color problem

$$Z = \{w, x, y, z\}$$

$$D_w = D_x = D_y = D_z = \{r, g, b\}$$

$$C = \{C_{\langle w,x \rangle}, C_{\langle w,y \rangle}, C_{\langle x,y \rangle}, C_{\langle x,z \rangle}, C_{\langle y,z \rangle}\}$$

where for each A and B

$$C_{\langle A,B \rangle} = \left\{ \begin{array}{ll} \{\langle A, r \rangle, \langle B, g \rangle\}, & \{\langle A, r \rangle, \langle B, b \rangle\}, \\ \{\langle A, g \rangle, \langle B, r \rangle\}, & \{\langle A, g \rangle, \langle B, b \rangle\}, \\ \{\langle A, b \rangle, \langle B, r \rangle\}, & \{\langle A, b \rangle, \langle B, g \rangle\} \end{array} \right\}$$