Constraint Programming

Introduction

Javier Larrosa

Department of Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Barcelona, Spain
larrosa@lsi.upc.edu

October 24, 2013

Outline

- Constraint Satisfaction Problems (CSP)
- Examples:
  - SAT
  - 8-queens
  - Graph Coloring
  - Knapsack
  - Combinatorial Auction
  - Round-Robin Tournament Scheduling
- Backtracking

Constraint Satisfaction Problem (CSP)

Example (Simple Numerical CSP)

$x, y, z \in \{3, 4, 5, 6, 7, 8\}$
$x + y = 10,\ x + z = 8,\ |y - z| = 2$
$x \leftarrow 4,\ y \leftarrow 6, z \leftarrow 4$ is one possible **solution**.

Many practical problems can be naturally modelled as CSPs
It is possible to design efficient general-purpose CSP solvers

Constraint Satisfaction Problem (CSP)

Definition

A **constraint satisfaction problem** is a tuple (X, D, C) where,

- $X = \{x_1, x_2, \ldots, x_n\}$ is the set of variables.
- $D = \{d_1, d_2, \ldots, d_n\}$ is the set of domains.
  - d_i is a finite set of potential values for x_i
- $C = \{c_1, c_2, \ldots, c_e\}$ is a set of constraints

Definition

A **solution** of a constraint satisfaction problem is a consistent complete assignment (i.e. it satisfies all the constraints)

Constraint Programming

Definition

A **constraint programming (CP)** system is a **modeling language** and a **solver**

- The modeling language must offer the possibility to model any CSP problem P.
- The solver must offer, at least, a functionality $Solve(P)$ which returns a solution of P if is there any.

Different CPs provide a variety of languages of different flavors and different expressive power

Different CPs provide a variety of solving tools. Some of them are quite rigid, while some others are very flexible.

In this course we will use the GECODE.

Example 0: Boolean Satisfiability

In the previous part of the course we addressed the following problem:

Definition

Let $\mathcal{F}$ be a CNF formula. The **SATisfiability** problem is the problem of finding a satisfying assignment (i.e, a model) for $\mathcal{F}$ if is there any.

This problem is a very special case of CSP where:

- only boolean variables are permitted
- only one type of constraints (*clauses*) is permitted

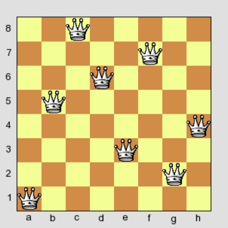
Example 0: Boolean Satisfiability

Note that each clause is a constraint that forbids exactly one partial assignment (how many complete assignments?)
Thus, SAT-solvers (such as Minisat) can be seen as CP systems where the language is CNF (very unexpressive, indeed) and the solving tool is the SAT-solver (very rigid, indeed).
Because they are so unexpressive and so rigid, they can be implemented very efficiently.

Example 1: 8-queens Problem (first version)

Definition

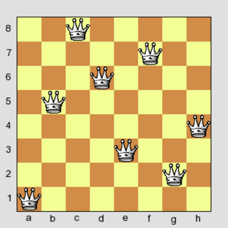
Place 8 queens in a 8×8 chess board in such a way that they do not attack each other



Example 1: 8-queens Problem (first version)

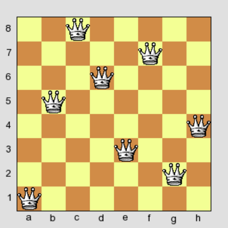
Definition

Place 8 queens in a 8×8 chess board in such a way that they do not attack each other



- **Variables/Domains:**
  - 8^2 boolean variables
  - x_{ij} refers to cell in row i, column j
  - $x_{ij} \in \{0, 1\}$ (there is a queen in row j).
- **Constraints:**
  - each row contains one queen
  - each column contains one queen
  - each diagonal contains at most one

Example 1: 8-queens Problem (first version)



- each row $i = 1..8$ contains one queen, $\sum_{j=1}^{8} x_{ij} = 1$
- each column $j = 1..8$ contains one queen, $\sum_{i=1}^{8} x_{ij} = 1$
- each diagonal contains at most one queen.
  - Each descending diagonal in the upper half $k = 1..8$,

$$\sum_{i=1}^{k} x_{i(i+8-k)} \leq 1$$

  - Each descending diagonal in the lower half $k = 1..8$,

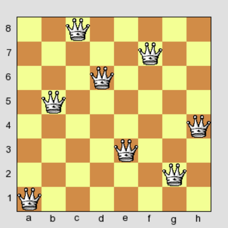
$$\sum_{i=1}^{k} x_{(i+8-k)i} \leq 1$$

  - Each ascending diagonal in the upper half ...
  - Each ascending diagonal in the lower half ...

Example 1: 8-queens Problem (second version)

Definition

Place 8 queens in a 8×8 chess board in such a way that they do not attack each other

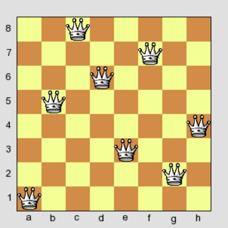


- **Variables/Domains:**
  - 8 variables with domain 1..8
  - x_i refers to the queen in row i
  - $x_i \leftarrow j$ (there is a queen in row i, column j).
- **Constraints:**

Example 1: 8-queens Problem (second version)

Definition

Place 8 queens in a 8×8 chess board in such a way that they do not attack each other

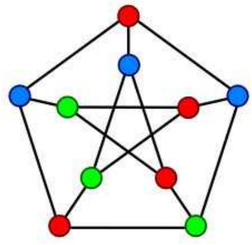


- **Variables/Domains:**
  - 8 variables with domain 1..8
  - x_i refers to the queen in row i
  - $x_i \leftarrow j$ (there is a queen in row i, column j).
- **Constraints:**
  - $\forall_{1 \leq i < j \leq 8}, \; x_i \neq x_j$
  - $\forall_{1 \leq i < j \leq 8}, \; |x_i - x_j| \neq |i - j|$

Example 2: Graph Coloring

Definition

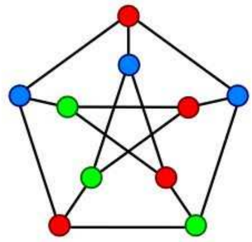
Given a graph $G = (V, E)$ and $k > 0$ colors, assign a color to each vertex in such a way that adjacent vertices have different colors



Example 2: Graph Coloring

Definition

Given a graph $G = (V, E)$ and $k > 0$ colors, assign a color to each vertex in such a way that adjacent vertices have different colors



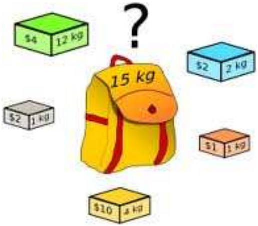
- **Variables/Domains:**
  - $|V|$ variables with domain $1..k$
  - x_i refers to vertex $i \in V$
  - $x_i \leftarrow j$ (vertex i gets color j)
- **Constraints:**

  $\forall_{(i,j) \in E}, \; x_i \neq x_j$

Example 3: Knapsack

Definition

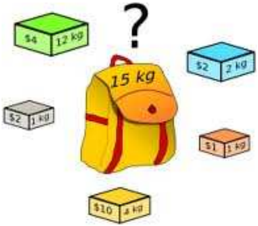
Given n items (each item i has a weight w_i and a value v_i), a capacity $W > 0$ and an integer $V > 0$, select a subset of the items such that their aggregated weight is less than W and aggregated value is more than V



Example 3: Knapsack

Definition

Given n items (each item i has a weight w_i and a value v_i), a capacity $W > 0$ and an integer $V > 0$, select a subset of the items such that their aggregated weight is less than W and aggregated value is more than V

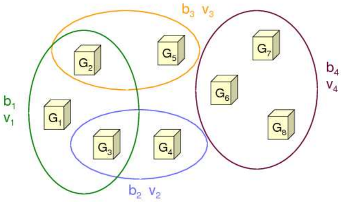


- **Variables/Domains:**
  - n boolean variables
  - x_i refers to item i
  - $x_i \in \{0, 1\}$ "item i is selected".
- **Constraints:**
  - $\sum_{i=1}^{n} w_i x_i < W$
  - $\sum_{i=1}^{n} v_i x_i > V$

Example 4: Combinatorial Auction

Definition

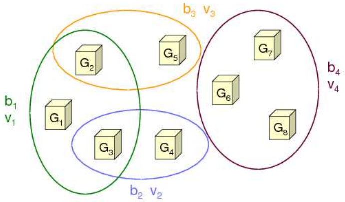
Consider a set of m goods, a set of n bids, where each bid i offers an amount of money v_i for a subset of the goods b_i, and a value $V > 0$. The goal is to find a subset of compatible bids with aggregated revenue higher than V. bidtaker revenue.



Example 4: Combinatorial Auction

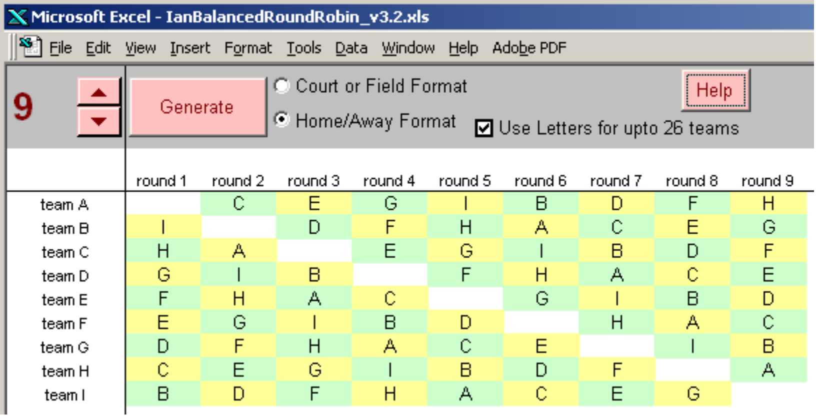
Definition

Consider a set of m goods, a set of n bids, where each bid i offers an amount of money v_i for a subset of the goods b_i, and a value $V > 0$. The goal is to find a subset of compatible bids with aggregated revenue higher than V. bidtaker revenue.



- **Variables/Domains:**
  - x_i refers to bid i
  - $x_i \in \{true, false\}$ "bid i is selected"
- **Constraints:**
  - $\forall_{i,j \texttt{ s.t. } b_i \cap b_j \neq \emptyset}, \; x_i + x_j \leq 1$
  - $\sum_{i=1}^{n} v_i x_i > V$

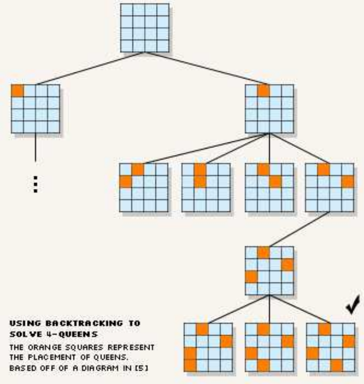
Example 5: Round-Robin Tournament



	round 1	round 2	round 3	round 4	round 5	round 6	round 7	round 8	round 9
team A		C	E	G	I	B	D	F	H
team B	I		D	F	H	A	C	E	G
team C	H	A		E	G	I	B	D	F
team D	G	I	B		F	H	A	C	E
team E	F	H	A	C		G	I	B	D
team F	E	G	I	B	D		H	A	C
team G	D	F	H	A	C	E		I	B
team H	C	E	G	I	B	D	F		A
team I	B	D	F	H	A	C	E	G	

Basic Backtracking Algorithm (BT)

Starting with an empty assignment, the algorithm tries at each step to extend it to one more variable, while preserving consistency.



- Depth-first traversal of a search tree
- Backtrack when *it is clear* that there is no solution below
- **Complexity:** $O(m^n \cdot e)$ (where n is the number of variables, m the size of the largest domain and e the number of constraints)
- In most problems m is a constant and e is a polinomial on n of fixed degree. Hence, the complexity is $O(2^n)$

Definition

Constraint Programming = Modeling + Propagation + Search