DM826 – Spring 2011

Modeling and Solving Constrained Optimization Problems

Lecture 2

Overview to Modelling and CP

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

[*Slides by Stefano Gualandi, Politecnico di Milano*]

Outline

1. Modelling
2. CP Overview
3. Modeling: Global Constraints

Resume

- First example: Send More Money
  first experience on modelling in MILP and CP
- SAT models
  - impose modelling rules (propositional calculus)
- MILP models
  - impose modelling rules: linear inequalities and objectives
  - emphasis on tightness and compactness of LP, strength of bounds (remove dominated constraints)
- CP models
  - a large variety of algorithms communicating with each other: global constraints
  - more expressiveness
  - emphasis on exploit substructres, include redundant constraints

Outline

1. Modelling
2. CP Overview
3. Modeling: Global Constraints

Second example: Sudoku

How can you solve the following Sudoku?

	4	3		8		2	5	
6								
					1		9	4
9					4		7	
			6		8			
	1		2					3
8	2		5					
								5
	3	4		9		7	1	

Sudoku: ILP model

Let y_{ijt} be equal to 1 if digit t appears in cell (i, j). Let N be the set $\{1, \ldots, 9\}$, and let J_{kl} be the set of cells (i, j) in the 3×3 square in position k, l.

$$\sum_{j \in N} y_{ijt} = 1, \qquad \forall i, t \in N,$$

$$\sum_{j \in N} y_{jit} = 1, \qquad \forall i, t \in N,$$

$$\sum_{i,j \in J_{kl}} y_{ijt} = 1, \qquad \forall k, l = \{1, 2, 3\}, t \in N,$$

$$\sum_{t \in N} y_{ijt} = 1, \qquad \forall i, j \in N,$$

$$y_{ija_t} = 1, \qquad \forall i, j \in \text{ given instance.}$$

Sudoku: CP model

$$
\begin{aligned}
&X_{ij} \in N, && \forall i, j \in N,\\
&X_{ij} = a_t, && \forall i, j \in \text{ given instance},\\
&\text{alldifferent}([X_{1i}, \ldots, X_{9i}]), && \forall i \in N,\\
&\text{alldifferent}([X_{i1}, \ldots, X_{i9}]), && \forall i \in N,\\
&\text{alldifferent}(\{X_{ij} \mid ij \in J_{kl}\}), && \forall k, l \in \{1, 2, 3\}.
\end{aligned}
$$

Sudoku: CP model (revisited)

$$X_{ij} \in N, \qquad \forall i, j \in N,$$
$$X_{ij} = a_t, \qquad \forall i, j \in \text{ given instance},$$
$$\text{alldifferent}([X_{1i}, \ldots, X_{9i}]), \qquad \forall i \in N,$$
$$\text{alldifferent}([X_{i1}, \ldots, X_{i9}]), \qquad \forall i \in N,$$
$$\text{alldifferent}(\{X_{ij} \mid ij \in J_{kl}\}), \qquad \forall k, l \in \{1, 2, 3\}.$$

Redundant Constraint:

$$\sum_{j \in N} X_{ij} = 45, \qquad \forall i \in N,$$
$$\sum_{j \in N} X_{ji} = 45, \qquad \forall i \in N,$$
$$\sum_{ij \in S_{\boldsymbol{kl}}} X_{ij} = 45, \qquad k, l \in \{1, 2, 3\}.$$

Outline

Constraint Reasoning

Combination

Simplification

Contradiction

Redundancy

General Purpose Algorithms

Search algorithms

organize and explore the search tree

- Search tree with branching factor at the top level nd and at the next level $(n-1)d$. The tree has $n! \cdot d^n$ leaves even if only d^n possible complete assignments.
- Insight: CSP is commutative in the order of application of any given set of action (the order of the assignment does not influence)
- Hence we can consider search algs that generate successors by considering possible assignments for only a single variable at each node in the search tree.
  The tree has d^n leaves.

Backtracking search

depth first search that chooses one variable at a time and backtracks when a variable has no legal values left to assign.

Backtrack Search

```
function BACKTRACKING-SEARCH(csp) returns a solution, or failure
  return RECURSIVE-BACKTRACKING({ }, csp)

function RECURSIVE-BACKTRACKING(assignment, csp) returns a solution, or failure
  if assignment is complete then return assignment
  var ← SELECT-UNASSIGNED-VARIABLE(VARIABLES[csp], assignment, csp)
  for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
     if value is consistent with assignment according to CONSTRAINTS[csp] then
        add {var = value} to assignment
        result ← RECURSIVE-BACKTRACKING(assignment, csp)
        if result ≠ failure then return result
        remove {var = value} from assignment
  return failure
```

Backtrack Search

- No need to copy solutions all the times but rather extensions and undo extensions
- Since CSP is standard then the alg is also standard and can use general purpose algorithms for initial state, successor function and goal test.
- Backtracking is uninformed and complete. Other search algorithms may use information in form of heuristics

General Purpose Backtracking

Implementation refinements

1) [Search] Which variable should we assign next, and in what order should its values be tried?

2) [Propagation] What are the implications of the current variable assignments for the other unassigned variables?

3) [Search] When a path fails – that is, a state is reached in which a variable has no legal values can the search avoid repeating this failure in subsequent paths?

Search

1) Which variable should we assign next, and in what order should its values be tried?

- Select-Initial-Unassigned-Variable
  degree heuristic (reduces the branching factor) also used as tied breaker
- Select-Unassigned-Variable
  Most constrained variable (DSATUR) = fail-first heuristic
  = Minimum remaining values (MRV) heuristic (speeds up pruning)
- Order-Domain-Values
  least-constraining-value heuristic (leaves maximum flexibility for subsequent variable assignments)

NB: If we search for all the solutions or a solution does not exists, then the ordering does not matter.

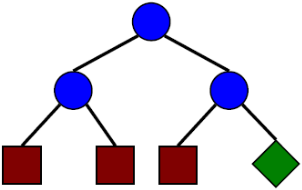
Search

Branching (aka, Labelling)

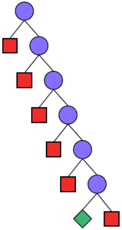
1. Pick a variable x with at least two values
2. Pick value v from $D(x)$
3. Branch with

$$x = v \qquad\qquad x \neq v$$
$$x < v \qquad\qquad x \geq v$$

The constraints for branching become part of the model in the subproblems generated



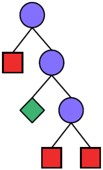
Lexicographic



SEND
+ MORE
= MONEY

9567
+ 1085
= 10652

First-fail



SEND
+ MORE
= MONEY

9567
+ 1085
= 10652

Constraint Propagation

2) What are the implications of the current variable assignments for the other unassigned variables?

Definition (Domain consistency)

A constraint C on the variables $X_1, \ldots, X_k$ is called domain consistent if for each variable X_i and each value $v_i \in D(X_i)$ $(i = 1, \ldots, k)$, there exist a value $v_j \in D(X_j)$ for all $j \neq i$ such that $(d_1, \ldots, d_k) \in C$.

Loose definition

Domain filtering is the removal of values from variable domains that are not consistent with an individual constraint.

Constraint propagation is the repeated application of all domain filtering of individual constraints until no domanin reduction is possible anymore.

Constraint Propagation

DEMO

- Forward checking
- Bounds consistency
- Domain consistency

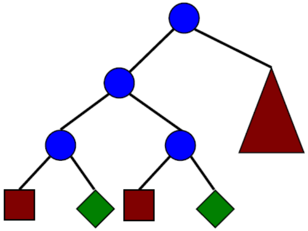
Constraint Propagation

Sudoku DEMO

Optimization Problems

Objective function to minimize $F(X_1, X_2, \ldots, X_n)$

- Naive approach: find all solutions and choose the best
- Branch and Bound approach
  - Solve a modified Constraint Satisfaction Problem by setting an (upper) bound z^* in the objective function



Dichotomic search: U upper bound, L lower bound $M = \frac{U+L}{2}$

Types of Variables and Values

- Discrete variables with finite domain:
  complete enumeration is $O(d^n)$
- Discrete variables with infinite domains:
  Impossible by complete enumeration.
  Propagation by reasoning on bounds.
  Eg, project planning.

  $$S_j + p_j \leq S_k$$

  NB: if only linear constraints, then integer linear programming
- Variables with continuous domains (time intervals)
  branch and reduce

  NB: if only linear constraints or convex functions then mathematical programming
- structured domains (eg, sets, graphs)

Outline

1. Modelling

2. CP Overview

3. Modeling: Global Constraints

Global Constraint: Sum

Sum constraint

Let $x_1, x_2, \ldots, x_n$ be variables. To each variable x_i, we associate a scalar $c_i \in \mathbb{Q}$. Furthermore, let z be a variable with domain $D(z) \subseteq \mathbb{Q}$. The sum constraint is defined as

$$\texttt{sum}([x_1, \ldots, x_n], z, c) = \left\{ (d_1, \ldots, d_n, d) \mid \forall i, d_i \in D(x_i), d \in D(z), d = \sum_{i=1,\ldots,n} c_i x_i \right\}.$$

Global Constraint: Knapsack

Knapsack constraint

Rather than constraining the sum to be a specific value, the knapsack constraint states the sum to be within a lower bound l and an upper bound u, i.e., such that $D(z) = [l, u]$. The knapsack constraint is defined as

$$\texttt{knapsack}([x_1, \ldots, x_n], z, c) =$$

$$\left\{(d_1, \ldots, d_n, d) \mid \forall i, d_i \in D(x_i), d \in D(z), d \leq \sum_{i=1,\ldots,n} c_i x_i\right\} \cap$$

$$\left\{(d_1, \ldots, d_n, d) \mid \forall i, d_i \in D(x_i), d \in D(z), d \geq \sum_{i=1,\ldots,n} c_i x_i\right\}.$$

CP Modeling Guidelines [Hooker, 2011]

1. A specially-structured subset of constraints should be replaced by a single global constraint that **captures the structure**, when a suitable one exists. This produces a more succinct model and can allow more effective filtering and propagation.

2. A global constraint should be replaced by a more specific one when possible, to exploit more effectively the special structure of the constraints.

3. The addition of redundant constraints (i..e, constraints that are implied by the other constraints) can improve propagation.

4. When two alternate formulations of a problem are available, including both (or parts of both) in the model may improve propagation. Different variables are linked through the use of channeling constraints.

References

Hooker J.N. (2011). **Hybrid modeling**. In *Hybrid Optimization*, edited by P.M. Pardalos, P. van Hentenryck, and M. Milano, vol. 45 of **Optimization and Its Applications**, pp. 11–62. Springer New York.

van Hoeve W. and Katriel I. (2006). **Global constraints**. In *Handbook of Constraint Programming*, chap. 6. Elsevier.