DM826 – Spring 2012

Modeling and Solving Constrained Optimization Problems

Lecture 2

Overview on CP

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

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

Outline

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 exploiting substructres, include redundant constraints

Outline

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.

$$
\begin{aligned}
&\sum_{j \in N} y_{ijt} = 1, && \forall i, t \in N, \\
&\sum_{j \in N} y_{jit} = 1, && \forall i, t \in N, \\
&\sum_{i,j \in J_{kl}} y_{ijt} = 1, && \forall k, l = \{1, 2, 3\}, t \in N, \\
&\sum_{t \in N} y_{ijt} = 1, && \forall i, j \in N, \\
&y_{ija_t} = 1, && \forall i, j \in \text{ given instance.}
\end{aligned}
$$

Sudoku: CP model

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

Sudoku: CP model (revisited)

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

Redundant Constraint:

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

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 final answer)
- 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.

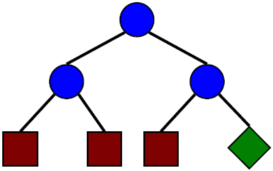
Search

Branching (aka, Labelling)

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

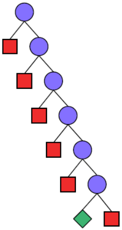
$$x = v \qquad\qquad x \neq v$$
$$x \leq v \qquad\qquad x > v$$

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



The inner nodes (blue circles) are choices, the red square leaf nodes are failures, and the green diamond leaf node is a solution.

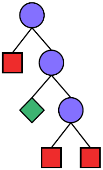
Lexicographic



```
   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 exists 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

Three consistency levels

Trade off between speed and propagation

- Forward checking
- Bounds consistency
- Domain consistency

Constraint Propagation

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

Problem shown as matrix

Each cell corresponds to a variable

Instantiated: Shows integer value (large)

Uninstantiated: Shows values in domain

Currently active constraint highlighted

Values removed at a step shown in blue

Values assigned at a step shown in red

`alldifferent` (`distinct`)

- Argument: list of variables
- Meaning: variables are pairwise different
- Reasoning: Forward Checking (FC)
    - When variable is assigned to value, remove the value from all other variables
    - If a variable has only one possible value, then it is assigned
    - If a variable has no possible values, then the constraint fails
    - Constraint is checked whenever one of its variables is assigned
    - Equivalent to decomposition into binary disequality constraints

Forward checking

H. Simonis' demo, slides 18-48

Can we do better?

Example:

$$\mathcal{P} = \langle X = (x, y, z), \mathcal{DE} = \{D(x) = \{1, 2\}, D(x) = \{1, 2\}, D(x) = \{1..3\}\},$$
$$\mathcal{C} = \{C_1 \equiv \texttt{alldiff}(x, y, z)\}\rangle$$

Bound Consistency

Example:
Idea (Hall Intervals)

- Take each interval of possible values, say size N
- Find all K variables whose domain is completely contained in interval
- If $K > N$ then the constraint is infeasible
- If $K = N$ then no other variable can use that interval
- Remove values from such variables if their bounds change
- If $K < N$ do nothing
- Re-check whenever domain bounds change

Bound Consistency

Definition

A constraint achieves bounds consistency, if for the lower and upper bound of every variable, it is possible to find values for all other variables between their lower and upper bounds which satisfy the constraint.

Can we do better?

- Bounds consistency only considers min/max bounds
- Ignores "holes" in domain
- Sometimes we can improve propagation looking at those holes

Example:

$$\mathcal{P} = \langle X = (x, y, z), \mathcal{DE} = \{D(x) = \{1, 3\}, D(x) = \{1, 3\}, D(x) = \{1..3\}\},$$
$$\mathcal{C} = \{C_1 \equiv \texttt{alldiff}(x, y, z)\}\rangle$$

Solutions and maximal matchings

- A Matching is subset of edges which do not coincide in any node
- No matching can have more edges than number of variables
- Every solution corresponds to a maximal matching and vice versa
- If a link does not belong to some maximal matching, then it can be removed

Domain Consistency

Definition

A constraint achieves domain consistency, if for every variable and for every value in its domain, it is possible to find values in the domains of all other variables which satisfy the constraint.

- Also called generalized arc consistency (GAC)
- or hyper arc consistency

Can we still do better?

- NO! This extracts all information from this one constraint
- We could perhaps improve speed, but not propagation
- But possible to use different model
- Or model interaction of multiple constraints

H. Simonis' demo, slides 80-142

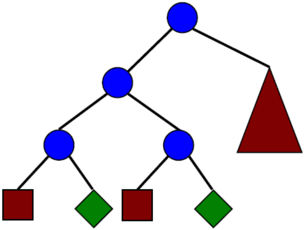
Typical?

- This does not always happen
- Sometimes, two methods produce same amount of propagation
- Possible to predict in certain special cases
- In general, tradeoff between speed and propagation
- Not always fastest to remove inconsistent values early
- But often required to find a solution at all

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)

References

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