Three short tutorials:

*Constraint programming,*
*Linear programming* and
*Knowledge-based systems.*

Philippe Morignot

IMARA Team – July 8, 2013
PART 1 ---
CONSTRAINT PROGRAMMING
Introduction

• **Constraint programming is one paradigm for modelling and solving combinatorial problems.**

• **Other algorithms:** Genetic algorithms, mixed integer programming, search algorithms in a state space (e.g., A*), simulated annealing, tabu search, ...

• A combinatorial problem is a problem defined by entities maintaining relations, to which one combination (a solution) must be found.

• One solution, several solutions, best solution.
Example (1/3)

• The sudoku game:

```
+---+---+---+
| 7 | 8 |   |
+---+---+---+
| 8 |   | 7  |
+---+---+---+
| 4 | 9 | 5  |
+---+---+---+
|   | 2 | 7  |
+---+---+---+
| 2 | 3 | 1  |
+---+---+---+
| 5 | 6 | 1  |
+---+---+---+
| 2 | 6 | 1  |
+---+---+---+
| 8 | 3 | 2  |
+---+---+---+
| 5 | 4 | 9  |
+---+---+---+
|   |   | 8  |
+---+---+---+
```
Example (2/3)

- N-queen problem (here, $N = 8$):
Example (3/3)

- Cryptarithmetic:

  UN
  + DEUX
  + DEUX
  + DEUX
  + DEUX
  ------------
  NEUF

  SEND
  + MORE
  + DEUX
  + DEUX
  + DEUX
  + DEUX
  + DEUX
  ------------
  MONEY
Model

• Variables with finite domain:
  – Variables: $V_i$
  – Domains: $D_j = \{ v_1, v_2, \ldots, v_{f(j)} \}$.
  – Forall $i$, $V_i \in D_i$

• Constraints:
  – For $k$ from 1 to $m$, $C_k = (X_k, R_k)$ avec :
    • $X_k = \{ V_{i1}, V_{i2}, \ldots, V_{ik} \}$ // The variables involved in constraint $C_k$
    • $R_k \subseteq D_{i1} \times D_{i2} \times \ldots \times D_{ik}$ // Possible values of these variables, together with constraint $C_k$

• A solution to a CSP (Constraint Satisfaction Problem) is a total consistant assignment.
Model / Example

• SEND + MORE = MONEY

• Variables:
  – S, M ∈ \{1, 2, 3, 4, 5, 6, 7, 8, 9\}
  – E, N, D, O, R, N, Y ∈ \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}

• Constraints:
  
  \[
  \begin{align*}
  D + E & = Y + 10 \times R_1 \\
  N + R + R_1 & = E + 10 \times R_2 \\
  E + O + R_2 & = N + 10 \times R_3 \\
  S + M + R_3 & = O + 10 \times M
  \end{align*}
  \]

  Addition carry variables : \(R_1, R_2, R_3 \in \{0, 1\}\)
Algorithm

- CHOOSE-UNASSIGNED-VARIABLE()
- SORT-DOMAIN-VALUES()
- Consistency check

- Backtracking: chronological, backjumping, conflict-directed
Filtering using Forward-Checking

Map coloring: set colors to regions A,B,C and D, while avoiding that two colors are adjacent. Possible colors are:

- A and B are green or red
- C is green, blue or red
- D is blue or green
Filtering using Forward-Checking

• Model:
Filtering using Forward-Checking

- Let us assume that B is set to the value RED.
Filtering using Forward-Checking

- Assignment of A to the value GREEN.
Filtering using Forward-Checking

- Assignment of C to the value BLUE.
Filtering using Forward-Checking

- Assignment of D to the value GREEN.

![Diagram showing the assignment process.](image-url)
Filtering using Forward-Checking

- Solution!

A ≠ B ≠ C ≠ D

{red} {green} {blue}

A B C D

A B C D

{green} {red} {blue}
Filtering using Forward-Checking

- But now let us assume that C is set to the value GREEN
Filtering using Forward-Checking

- Assignment of D to BLUE, and of A and B to RED.
Filtering using Forward-Checking

- Removal of RED from B’s domain.
Filtering using Forward-Checking

- The domain of B is empty: FAILURE!
  (and then backtracking is required...)
Consistency

• **Definition:** A CSP is $k$-consistent iff, for all subset of $(k - 1)$ variables, and for all assignment of these variables, a consistent value can always be assigned to the $k$-th variable.

• **Example:**
  – 1-consistency = node consistency
  – 2-consistency = arc consistency
  – 3-consistency = path consistency
Arc-consistency is not enough: path consistency is required, e.g., above.

- Strong k-consistency.
- Algorithms AC1 to AC8.
Conclusion – Part 1

• Constraint programming is a paradigm for modelling and solving combinatorial problems.


• OPL Studio from IBM (ex-ILOG):

• French Association for Constraint Programming:
  http://afpc.greyc.fr/web/
PART 2 ---
LINEAR PROGRAMMING
Model

- **Algebraic notation:** \( n \) variables, \( m \) inequalities.

Min \( \sum_j c_j x_j \) such that

\[
\forall i, \sum_j a_{i,j} x_j \geq b_i \quad \text{//} \leq \quad \text{when Max}
\]

\[
\forall j, x_j \geq 0
\]

- **Matrix notation:**

Min \( tC X \) such that

\[
AX \geq B
\]

\[
X \geq 0
\]
The Simplex Algorithm: Example

- A factory can build 3 products with production rates of 50, 25 and 75 units/hour, respectively.
- Machines can be used for 45 hours at most.
- Sales benefits of products are 4€, 12€ and 3€, respectively.
- Customers ask for 1000, 500 and 1500 units, respectively.
Example: Model (1/2)

• **Variables:**
  \[ x_i = \text{number of units of product } i \text{ built by the factory.} \]
  \[ i = 1, 2, 3 \text{ et } x_i \geq 0 \]

• **Objective function:**
  \[ z = \max \left( 4 x_1 + 12 x_2 + 3 x_3 \right) \]

• **Constraints:**
  \[
  \begin{align*}
  \frac{x_1}{50} + \frac{x_2}{25} + \frac{x_3}{75} & \leq 45 \\
  x_1 & \leq 1000 \\
  x_2 & \leq 500 \\
  x_3 & \leq 1500 \\
  x_1, x_2, x_3 & \geq 0
  \end{align*}
  \]
Example: Model (2/2)

- **Linear program:**

  \[
  \text{Max} \ ( 4x_1 + 12x_2 + 3x_3 ) \quad \text{such that}
  \]

  \[
  \begin{align*}
  x_1 & \leq 1000 \quad (1) \\
  x_2 & \leq 500 \quad (2) \\
  x_3 & \leq 1500 \quad (3) \\
  3x_1 + 6x_2 + 2x_3 & \leq 6750 \quad (4) \\
  x_1, x_2, x_3 & \geq 0
  \end{align*}
  \]
Example: Geometric model

- $z_0 = 0$
- $z_A = 4000$
- $z_B = 7000$
- $z_C = 6000$
- $z_D = 4500$
- $z_E = 8500$
- $z_G = 10500$

P(1000, 125, 15000) et $z_P = 10000$
Q(250, 500, 1500) et $z_Q = 11500$
R(1000, 500, 375) et $z_R = 10750$
Example: The simplex algorithm

Points: O, C, B, R, Q.

- 10 vertices to explore
- 4 iterations for the simplex algorithm to find the solution.
Mixed integer programming

• The simplex algorithm finds a solution in \( \mathbb{R} \)
• What if the variable are integers?
  \((x_1, x_2, \ldots, x_n) \in \mathbb{N} \) instead of \( \mathbb{R} \)
• ... or are boolean? \((x_1, x_2, \ldots, x_n) \in \{0, 1\} \) instead of \( \mathbb{R} \)

• Branch & bound algorithm.
  – Heuristic search in a tree.
  – A node includes the (relaxed) solution on \( \mathbb{R} \) and additional constraints.
Example (1/6)

• Let \((P)\) be the problem:

\[
\begin{align*}
\text{max} \ (4x_1 - x_2) & \text{ such that } \\
7x_1 - 2x_2 & \leq 14 \\
x_2 & \leq 3 \\
2x_1 - 2x_2 & \leq 3 \\
x_1, x_2 & \geq 0 \\
(x_1, x_2) & \in \mathbb{N}
\end{align*}
\]

• Let \((P')\) be the problem \((P)\) but with variables in \(\mathbb{R}\)
Branch & bound (2/6)

\( (P) : \emptyset \)
\( (P') : x' = \frac{20}{7} ; 3 \)
\( z' = \frac{59}{7} \), donc \( z \leq 8 \)
Branch & bound (3/6)

(P ) : $x_1 \leq 2$
(P' ) : $x' = (2 ; 1/2)$
$z' = 15/2$, donc $z \leq 7$

(P ) : $x_1 \geq 3$
(P' ) : $\emptyset$

(P ) : $\emptyset$
(P' ) : $x' = (20/7 ; 3)$
$z' = 59/7$, donc $z \leq 8$

$x_1 \leq 20/7$
$x_1 \geq 20/7$
Branch & bound (4/6)

(P) : \( x_1 \leq 2 \)

(P') : \( x' = (20/7; 3) \)

\( z' = 59/7 \), donc \( z \leq 8 \)

(P) : \( x_1 \geq 3 \)

(P') : \( \emptyset \)

\( x_1 \leq 20/7 \)

\( x_1 \geq 20/7 \)

(x_2 \leq 1/2)

(P) : \( x_1 \leq 2, x_2 \leq 0 \)

(P') : \( x' = (3/2; 0) \)

\( z' = 6 \), donc \( z \leq 6 \)

(x_2 \geq 1/2)

(P) : \( x_1 \leq 2, x_2 \geq 1 \)

(P') : \( x' = (2; 1) \)

\( z' = 7 \), donc \( z \leq 7 \)
Branch & bound (5/6)

\[(P) : x_1 \leq 2, x_2 \leq 0 \]
\[(P') : x' = (3/2 ; 0) \]
\[z' = 6, donc z \leq 6 \]

\[x_2 \leq 1/2 \]

\[(P) : x_1 \leq 2, x_2 \leq 0 \]
\[(P') : x' = (3/2 ; 0) \]
\[z' = 6, donc z \leq 6 \]

\[x_1 \geq 3/2 \]

\[(P) : x_1 \leq 2, x_2 \geq 1 \]
\[(P') : x' = (2 ; 1) \]
\[z' = 7, donc z \leq 7 \]

\[x_2 \geq 1/2 \]

\[(P) : x_1 \geq 3 \]
\[(P') : \emptyset \]

\[x_1 \leq 20/7 \]

\[(P) : \emptyset \]
\[z' = 59/7, donc z \leq 8 \]

\[x_1 \geq 20/7 \]

\[(P) : x_1 \geq 3 \]
\[(P') : \emptyset \]

\[x_1 \leq 3/2 \]

\[(P) : x_1 \leq 1, x_2 \leq 0 \]
\[(P') : x' = (1 ; 0) \]
\[z' = 4, donc z \leq 4 \]

\[S_1 = (1 ; 0 ; z = 4) \]

\[x_1 \geq 3/2 \]

\[(P) : x_1 = 2, x_2 = 0 \]
\[(P') : \emptyset \]
Branch & bound (6/6)

\((P)\) : \(x_1 \leq 2\)
\((P')\) : \(x' = (20/7 ; 3)\)
\(z' = 59/7, \text{ donc } z \leq 8\)

\((P)\) : \(x_1 \geq 3\)
\((P')\) : \(\emptyset\)

\((P)\) : \(x_1 \leq 2\)
\((P')\) : \(x' = (2 ; 1/2)\)
\(z' = 15/2, \text{ donc } z \leq 7\)

\((P)\) : \(x_1 \geq 3\)
\((P')\) : \(\emptyset\)

\((P)\) : \(x_1 \leq 2, x_2 \leq 0\)
\((P')\) : \(x' = (3/2 ; 0)\)
\(z' = 6, \text{ donc } z \leq 6\)

\((P)\) : \(x_1 \leq 2, x_2 \geq 1\)
\((P')\) : \(x' = (2 ; 1)\)
\(z' = 7, \text{ donc } z \leq 7\)

\(S_2 = (2 ; 1 ; z = 7)\)

\((P)\) : \(x_1 \leq 1, x_2 \leq 0\)
\((P')\) : \(x' = (1 ; 0)\)
\(z' = 4, \text{ donc } z \leq 4\)

\(S_1 = (1 ; 0 ; z = 4)\)
Typical problems

• **Transportation problem:** Several customers, several depots; A customer can be delivered by several depots; Each depot has a stock capacity; Each customer asks for some quantity; Carrying one unit of product from a depot to a customer has a price.

• **Variants:** ... with a multiple of N products; ... a customer is assigned to a **unique** depot; ... or a depot can be open/closed with an opening cost;
Modelling tricks

- Goal programming
- Semi-continuous variable
- IF ... THEN ...
- Disjunction
- Absolute value (inside cost, inside constraints)
- Quadratic cost
- Step-wise cost
Duality

• Primal lin. prog.:
  Min $C x$ tel que :
  $A x \geq B$
  $x \geq 0$

• Dual lin. prog.:
  Max $^t B y$ tel que :
  $^t A y \leq ^t C$
  $y \geq 0$
Column generation algorithm

- **Master problem:**

  - From 1000 to 10 000 variables
  - $x_j$
  - $c_j$
  - $a_{ij} \geq b_i$

- **Sub-problem:**

  - $> 10 000$ variables.
  - Created on demand.
  - $y_i$

**WHILE** $\exists j / \Delta_j = c_j - \sum_i a_{i,j}y_i < 0$
Multi-criteria optimization

- **LP single-criterion:**

  
  \[
  \min \left( \sum_j c_j x_j \right) \\
  \text{such that} \\
  \forall i, \ \sum_j a_{i,j} x_j \geq b_i \\
  \forall j, \ x_j \geq 0
  \]

- **LP multi-criteria:**

  \[
  \forall k, \ \min \left( F_k = \sum_j c_{j,k} x_j \right) \\
  \text{such that} \\
  \forall i, \ \sum_j a_{i,j} x_j \geq b_i \\
  \forall j, \ x_j \geq 0
  \]
Multi-criteria optimization: Example

- In a problem of human resource planning, maximize profit, the social relationships and the quality of services.

- Criteria are inter-dependent: how to find an « optimal » solution?
Multi-criteria optimization

- **Definition**: for a LP of minimization, solution $x$ dominates solution $y$ iff
  $$x D y \iff \forall k, f_k(x) \leq f_k(y)$$

![Diagram 1](x dominates y)

![Diagram 2](x does not dominate y)
Multi-criteria optimization

- Upper right quadrant: solutions dominated by $s_1$.
- Pareto front = \{ $s_1$, $s_2$, ..., $s_{10}$ \}  // efficient solutions
- Which solution on the Pareto front is the « best » one?
« Best » solution? (1/2)

• **Criteria aggregation method:**

\[ f(x) = p_1 f_1(x) + p_2 f_2(x) + \ldots + p_n f_n(x) \]

\[ \min f(x) \]

– Simple but (i) does not necessarily reach an optimum; And (ii) how to set the \( p_i \)?

• **Serialization method:**

*Set an order on criteria, and solve the problem along this order.*

– Efficient, problem decomposition, but (i) n optimization to compute; And (ii) the optimum changes along the criterion order.
• **Gain array method:** Separately solve along every criterion, and minimize the maximum deviation from the ideal point \((F_{11}, F_{22}, ..., F_{kk})\).

<table>
<thead>
<tr>
<th>Criterion</th>
<th>(F_1)</th>
<th>(F_2)</th>
<th>(\ldots)</th>
<th>(F_k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solution 1</td>
<td>(F_{11})</td>
<td>(F_{12})</td>
<td>(\ldots)</td>
<td>(F_{1k})</td>
</tr>
<tr>
<td>Solution 2</td>
<td>(F_{21})</td>
<td>(F_{22})</td>
<td>(\ldots)</td>
<td>(F_{2k})</td>
</tr>
<tr>
<td>\ldots</td>
<td>(F_{k1})</td>
<td>(F_{k2})</td>
<td>(\ldots)</td>
<td>(F_{kk})</td>
</tr>
</tbody>
</table>

• Min \(\lambda\) such that

\[
\begin{align*}
\forall k, F_k - F_{kk} & \leq \lambda \left( \max_i (F_{ik}) - F_{kk} \right) \quad // \ 0 \leq \lambda \leq 1 \\
\forall k, F_k & = \sum_j c_{j,k} x_j \\
\forall j, x_j & \geq 0
\end{align*}
\]
Conclusion – Part 2

• With the simplex algorithm, linear programming finds a solution in $\mathbb{R}$ which minimizes a cost function, while respecting linear inequalities.

• With the branch & bound method, mixed integer programming does the same thing but on $\mathbb{N}$.


• CPLEX from IBM (ex-ILOG)
  
PART 3 --- KNOWLEDGE-BASED SYSTEMS
Expert systems

Expert system

Knowledge base

Rule base

Fact base

Inference engine

Expert

User
Inference Rules

• Rule base:

(defrule is-uncle
  (<x> ^father-of <y>) (<z> ^sex male) (<z> ^brother-of <x>)
  => { <z> ^uncle-of <y> }
)

• Fact base:

(fact Louis ^sex male)
(fact Louis ^brother-of Alexandre)
(fact Alexandre ^father-of Jean)

• Inferred fact: (add-fact Louis ^uncle-of Jean)
Forward vs. Backward Chaining

- **Rule base:**

- **Fact base:** \{A, B, C, D, E, X\}

- **Questions:**
  - Forward: what can be inferred?
  - Backward: can we prove Z? And N?
Inference Engine

• **Procedure FORWARD-CHAINING** (Rules, Facts)
  WHILE there remains rules in Rules to fire
    1. Lookup for rules of Rules which could fire (pattern-matching)
    2. Solve conflicts to choose one candidate rule R
    3. Fire R (inference)

• **Function BACKWARD-CHAINING** (Goal, Rules, Facts)
  1. Lookup for rules of Rules which conclude on Goal
  2. IF one such rule can fire with Facts, THEN fire it. SUCCESS
  3. Solve conflicts to determine one candidate rule R
  4. AND [BACKWARD-CHAINING(LHS(R), Rules, Facts)]
Rete algorithm

- **Rule base:**
  - $P_1 : C_1 \land C_2 \land C_3$
  - $P_2 : C_1 \land C_2 \land C_4 \land C_5$
  - $P_3 : C_1 \land C_2 \land C_4 \land C_3$
Conclusion – Part 3

• Knowledge-based system = (rule base + fact base) + inference engine.

• Seminal papers:
  – B. Buchanan, E. Shortliffe. [MYCIN]

• JESS from Sandia National Laboratories. http://herzberg.ca.sandia.gov/
THANK YOU!