Flexible Optimization: Nurse Scheduling with Constraint Programming and Automata

Sophie Demassey

Centre de Mathématiques Appliquées, MINES ParisTech
http://sofdem.github.io/

CMP, Gardanne, 3 July 2014
mutability of practical recurring problems

example 1: online data center resource management

http://btrp.inria.fr/ [Hermenier09]
mutability of practical recurring problems

example 2: employee timetabling

https://github.com/sofdem/chocoETP [Menana09]
outline

1. Mutable Problem
   - Nurse Scheduling

2. Flexible Tools
   - finite automata
   - global constraints

3. Flexible Solutions
   - multicost-regular = automata + global constraints
   - ChocoETP = automata + CP + local search

4. Conclusion
Nurse Scheduling Problem
an illustration of mutability
Nurse Scheduling Problem

- $I$ set of nurses
- $T$ discrete time horizon  \[28 \text{ days}\]
- $A$ set of activities $N \text{ night, } M \text{ morning, } E \text{ evening, } R \text{ rest}$
Nurse Scheduling Problem

- $I$ set of nurses
- $T$ discrete time horizon
- $A$ set of activities
- cover constraints $C_t$ / day $t$
- working rules $R_i$ / nurse $i$

- $28$ days
- $N$ night, $M$ morning, $E$ evening, $R$ rest
- between 2 and 3 nurses at night
- at least 2 mornings a week
Nurse Scheduling Problem

- $I$ set of nurses
- $T$ discrete time horizon
- $A$ set of activities
- cover constraints $C_t / \text{day } t$
- working rules $R_i / \text{nurse } i$

*28 days*

$N \text{ night, } M \text{ morning, } E \text{ evening, } R \text{ rest}$

between 2 and 3 nurses at night

at least 2 mornings a week
working rules

Examples:

- *between 2 and 3 rests every 7 days*
- *no 3 consecutive nights a week*
- *a rest and a night every week-end*

**mutable, heterogeneous, hard/soft**
Examples:

- *between 2 and 3 rests every 7 days*
- *no 3 consecutive nights a week*
- *a rest and a night every week-end*

**mutable, heterogeneous, hard/soft**
working rules

Examples:

- *between 2 and 3 rests every 7 days*
- *no 3 consecutive nights a week*
- *a rest and a night every week-end*

**mutable, heterogeneous, hard/soft**

<table>
<thead>
<tr>
<th>how ?</th>
<th>forbid</th>
<th>enforce</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>what ?</td>
<td>activity</td>
<td>stretch</td>
<td>pattern</td>
</tr>
<tr>
<td>when ?</td>
<td>fixed time</td>
<td>sliding period</td>
<td>fixed period</td>
</tr>
</tbody>
</table>
working rules

Examples:

- *between 2 and 3 rests every 7 days*
- *no 3 consecutive nights a week*
- *a rest and a night every week-end*

<table>
<thead>
<tr>
<th>how?</th>
<th>forbid</th>
<th>enforce</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>what?</td>
<td>activity</td>
<td>stretch</td>
<td>pattern</td>
</tr>
<tr>
<td>when?</td>
<td>fixed time</td>
<td>sliding period</td>
<td>fixed period</td>
</tr>
</tbody>
</table>
working rules

Examples:

- between 2 and 3 rests every 7 days
- no 3 consecutive nights a week
- a rest and a night every week-end

mutable, heterogeneous, hard/soft

<table>
<thead>
<tr>
<th>how?</th>
<th>forbid</th>
<th>enforce</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>what?</td>
<td>activity</td>
<td>stretch</td>
<td>pattern</td>
</tr>
<tr>
<td>when?</td>
<td>fixed time</td>
<td>sliding period</td>
<td>fixed period</td>
</tr>
</tbody>
</table>
working rules

Examples:

- *between 2 and 3 rests every 7 days*
- *no 3 consecutive nights a week*
- *a rest and a night every week-end*

mutable, heterogeneous, hard/soft
working rules

Examples:
- *between 2 and 3 rests every 7 days*
- *no 3 consecutive nights a week*
- *a rest and a night every week-end*

mutable, heterogeneous, hard/soft

individual constraint penalties (to minimize)

ex: \( 5 \times \text{occurrence(violation)}^2 \)
working rules

Examples:

- *between 2 and 3 rests every 7 days*
- *no 3 consecutive nights a week*
- *a rest and a night every week-end*

mutable, heterogeneous, hard/soft

⇒ *high-level modelisation tools*
⇒ *auto-configurable algorithms*
Flexible tools in Combinatorial Optimization
finite automata
flexible tool #1
### formal languages

- **alphabet**: $\Sigma$ a finite non-empty set of symbols
  \[ \{a, b\} \]

- **word/string**: $w \in \Sigma^n$ a finite sequence of symbols
  \[ aaabb \]

- **language**: $L \subseteq \Sigma^*$ a set of words
  \[ \{ab, ba, aab, bba, aaab, bbba, \ldots\} \]
formal languages

- **alphabet:** $\Sigma$ a finite non-empty set of symbols 
  \{a, b\}
- **word/string:** $w \in \Sigma^n$ a finite sequence of symbols
  aaabb
- **language:** $L \subseteq \Sigma^*$ a set of words
  \{ab, ba, aab, bba, aaab, bbba, \ldots\}

- **classes and recognizers:** regular, context-free, etc.
formal languages

- **alphabet**: $\Sigma$ a finite non-empty set of symbols
  $$\{a, b\}$$

- **word/string**: $w \in \Sigma^n$ a finite sequence of symbols
  $$aaabb$$

- **language**: $\mathcal{L} \subseteq \Sigma^*$ a set of words
  $$\{ab, ba, aab, bba, aaab, bbba, \ldots\}$$

- **classes and recognizers**: regular, context-free, etc.

- **operations**: union, concatenation, closure, etc.
formal languages

- **alphabet**: $\Sigma$ a finite non-empty set of symbols
  
  \{a, b\}

- **word/string**: $w \in \Sigma^n$ a finite sequence of symbols
  
  $aaabb$

- **language**: $L \subseteq \Sigma^*$ a set of words
  
  \{ab, ba, aab, bba, aaab, bbba, …\}

- **classes and recognizers**: regular, context-free, etc.

- **operations**: union, concatenation, closure, etc.

- **properties**: emptiness, membership, universality, etc.
generators and recognizers

\[ \mathcal{L} = \{ab, ba, aab, bba, aaab, bbba, \ldots\} \]

1 infinite **regular** language, 3 finite representations:

---

**finite automaton**

![Finite Automaton Diagram]

**regular expression**

\[(a+b)|((b^+)a)\]

**formal grammar**

\[
\begin{align*}
S & \rightarrow aA | bB \\
A & \rightarrow aA | b \\
B & \rightarrow bB | a
\end{align*}
\]
what purpose?

- implicit and concise (finite) representation
- human-readable and machine-processable
- theories and algorithms for operations and decision properties
- models of discrete systems like languages, protocols
what purpose?

- implicit and concise (finite) representation
- human-readable and machine-processable
- theories and algorithms for operations and decision properties
- models of discrete systems like languages, protocols
- models of working rules
  - alphabet: set of activities $A = \{M, E, N, R\}$
  - word: $w \in A^T$ schedule of an employee
  - language: constrained set of schedules
working rules as a language
working rules as a language

- rule $R$ as a regexp $E_R$ [Pesant04]

  no more than 2 consecutive nights: $E_R = \neg(\text{NNN})$
working rules as a language

- rule \( R \) as a regexp \( E_R \) [Pesant04]

  no more than 2 consecutive nights: \( E_R = \neg (NNN) \)

- feasible schedules as a regular language \( \mathcal{L}^R \cap A^T \) with

\[
\mathcal{L}^R = \bigcap_{R \in \mathcal{R}} \mathcal{L}(E_R) = \mathcal{L}(\neg \bigcup_{R \in \mathcal{R}} \neg E_R)
\]
working rules as a language

- rule $R$ as a regexp $E_R$ [Pesant04]
  
  no more than 2 consecutive nights: $E_R = \neg(\text{NNNN})$

- feasible schedules as a regular language $\mathcal{L}^R \cap \mathcal{A}^T$ with
  
  $$\mathcal{L}^R = \bigcap_{R \in \mathcal{R}} \mathcal{L}(E_R) = \mathcal{L}(\neg \bigcup_{R \in \mathcal{R}} \neg E_R)$$

- extension to context-free grammars [Sellman06, Quimper06, Côté10]
  
  $$\mathcal{L}(S \rightarrow \epsilon, S \rightarrow aSb) = \{a^n b^n \mid n \in \mathbb{N}\}$$
working rules as a language

- rule $R$ as a regexp $E_R$ [Pesant04]

  no more than 2 consecutive nights: $E_R = \neg(NNN)$

- feasible schedules as a regular language $\mathcal{L}^R \cap A^T$ with

  $$\mathcal{L}^R = \bigcap_{R \in \mathcal{R}} \mathcal{L}(E_R) = \mathcal{L}(\neg \bigcup_{R \in \mathcal{R}} \neg E_R)$$

- extension to context-free grammars [Sellman06, Quimper06, Côté10]

  $$\mathcal{L}(S \to \epsilon, S \to aSb) = \{a^n b^n \mid n \in \mathbb{N}\}$$

- extension to weighted automata [Demassey05, Menana09]

  for counting, optimization and soft rules
**weighted automata**

Transition costs, path cost, and bounds

- Add a vector of costs (index dependent) to each transition.
- The cost of the word is the sum of the transition costs.
- Restrict the language to words with costs within given bounds.

![Diagram of a weighted automaton with transitions labeled with costs and final states labeled with constraints.](image-url)
working rules as weighted automata \cite{Menana09}

### automated modeling tool in ChocoETP

1. model each rule including penalties as a language
   \[ \Rightarrow \text{regex or weighted automaton} \]
2. compute the language intersection
   \[ \Rightarrow \text{multi-weighted automaton} \]
working rules as weighted automata [Menana09]

automated modeling tool in ChocoETP

1. model each rule including penalties as a language
   \[\Rightarrow \text{regex or weighted automaton}\]

2. compute the language intersection
   \[\Rightarrow \text{multi-weighted automaton}\]

include parsers for different benchmark formats:

- ASAP3 (XML) [www.staffrostersolutions.com]
- NRP10 (XML) [www.kuleuven-kortrijk.be]
- NSPLib (csv) [www.projectmanagement.ugent.be]
- ETPShoe (csv+txt) [Demassey05]
modeling rules (ex: activity count)

at least one rest on week # 2

- hard rule, 2 alternatives:
  - a regexp $A\{7\}((\neg R)\ast RA\ast)A\{14\}$
  - or $A\ast$ with a counter $Z \in [1, 28]$ and $c_{tR} = 1$ iff $t \in [8, 14]$
modeling rules (ex: activity count)

at least one rest on week # 2

- **hard rule**, 2 alternatives:
  - a regexp $A^7(\neg R)^* R A^* A^{14}$
  - or $A^*$ with a counter $Z \in [1, 28]$ and $c_{tR} = 1$ iff $t \in [8, 14]$
- **soft rule**: (ex: fixed penalty of 10 if no rest on week 2)
  - $A^*$ with a counter $Z \in [0, 28]$ with $c_{tR} = 1$ iff $t \in [8, 14]$
  - and an external cost $Y \in [0, 10]$ with $Y = 10 \iff Z < 1$
modeling rules (ex: sliding stretch)

*between 3 and 5 consecutive night shifts*

- **hard rule:**
modeling rules (ex: sliding stretch)

between 3 and 5 consecutive night shifts

- hard rule:

- soft rule: \((\text{hard bounds}[0, 7] \text{ and quadratic penalty})\)

with a cost/counter \(Y = Z \in [0, +\infty]\)
modeling rules (ex: forbid pattern)

*at least one rest after 2 consecutive night shifts*

- hard rule:
  - $\neg (A^* (\neg N (\neg R)) A^*)$

Flexible optimization Flexible Tools finite automata
modeling rules (ex: forbid pattern)

- hard rule:
  - $\neg (A^* (NN(\neg R))A^*)$

- soft rule: (ex: linear penalty)

1. build the DFA corresponding to $(A^* (NN(\neg R)\beta^*)^*)^*$
2. get $Q_\beta$ the set of states $q$ with outgoing transition $\beta$
3. add a cost $c = 1$ on every ingoing transition of $Q_\beta$
4. associate a cost/counter $Y = Z \in [0, +\infty]$
aggregating rules

satisfying a conjunction of rules

- $R^1 \land R^2$ holds iff
  
  $X \in \mathcal{L}(\Pi^1) \cap \mathcal{L}(\Pi^2) \land Z^1 = \sum_t c^1_t X_t \land Z^2 = \sum_t c^2_t X_t$
aggregating rules

satisfying a conjunction of rules

- \( R^1 \land R^2 \) holds iff
  \[
  X \in \mathcal{L}(\Pi^1) \cap \mathcal{L}(\Pi^2) \land Z^1 = \sum_t c^1_t X_t \land Z^2 = \sum_t c^2_t X_t
  \]
- WFA intersection in the tropical semiring of higher dimension:
  \[
  (\Pi^1, [c^1, 0]) \cap (\Pi^2, [0, c^2]) \in WFA(\Sigma, \mathbb{R}^{n_1+n_2})
  \]
## aggregating rules

### satisfying a conjunction of rules

- **$R^1 \land R^2$ holds iff**

\[
X \in \mathcal{L}(\Pi^1) \cap \mathcal{L}(\Pi^2) \land Z^1 = \sum_t c^1_{tX_t} \land Z^2 = \sum_t c^2_{tX_t}
\]

- **WFA intersection in the tropical semiring of higher dimension:**

\[
(\Pi^1, [c^1, 0]) \cap (\Pi^2, [0, c^2]) \in WFA(\Sigma, \mathbb{R}^{n_1+n_2})
\]
aggregating rules

<table>
<thead>
<tr>
<th>satisfying a conjunction of rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R^1 \land R^2$ holds iff</td>
</tr>
<tr>
<td>$X \in \mathcal{L}(\Pi^1) \cap \mathcal{L}(\Pi^2) \land Z^1 = \sum_t c^1_t X_t \land Z^2 = \sum_t c^2_t X_t$</td>
</tr>
<tr>
<td>WFA intersection in the tropical semiring of higher dimension:</td>
</tr>
<tr>
<td>$(\Pi^1, [c^1, 0]) \cap (\Pi^2, [0, c^2]) \in WFA(\Sigma, \mathbb{R}^{n_1+n_2})$</td>
</tr>
</tbody>
</table>

(our) intersection algorithm in $WFA(\Sigma, \mathbb{R}^n)$

| convert $WFA(\Sigma, \mathbb{R}^n)$ to $FA(\Sigma \times \mathbb{R}^n)$ and naive intersection modified: $((q_1, q_2), (\sigma_1, \sigma_2), (q'_1, q'_2)) \in \Delta \leftrightarrow$ |
| $(q_1, \sigma_1, q'_1) \in \Delta_1 \land (q_2, \sigma_2, q'_2) \in \Delta_2 \land \text{symbol}(\sigma_1) = \text{symbol}(\sigma_2)$ |
global constraints
flexible tool #2
constraint satisfaction problem (CSP)

A solution:

\[(x_1, \ldots, x_n) \in D_1 \times \cdots \times D_n \text{ s.t. } C_j(x_1, \ldots, x_n) \text{ holds } \forall j = 1, \ldots, m\]
constraint satisfaction problem (CSP)

A set of variables $X_1, X_2, \ldots, X_n$
on finite (discrete) domains $D_1, D_2, \ldots, D_n$
related by constraints $C_1, \ldots, C_m$

A solution:

$$(x_1, \ldots, x_n) \in D_1 \times \cdots \times D_n \text{ s.t.}$$

$C_j(x_1, \ldots, x_n) \text{ holds } \forall j = 1, \ldots, m$$
sudoku as a CSP

\[ X_0, X_1, \ldots, X_{80} \]

\[ D_i = [0, 9] \quad \forall i \in [0, 80] \]

\[ X_0 = 2, \; X_1 = 6, \; \ldots \]

\[ X_i \neq X_j \quad \forall (i, j) \in L \]

\[ X_i \neq X_j \quad \forall (i, j) \in C \]

\[ X_i \neq X_j \quad \forall (i, j) \in S \]

credit: N. Jussien
sudoku as a CSP

\[ X_0, X_1, \ldots, X_{80} \]
\[ D_i = [0, 9] \quad \forall i \in [0, 80] \]
\[ X_0 = 2, \; X_1 = 6, \ldots \]
\[ X_i \neq X_j \quad \forall (i, j) \in L \]
\[ X_i \neq X_j \quad \forall (i, j) \in C \]
\[ X_i \neq X_j \quad \forall (i, j) \in S \]

arc consistency of \( X_0 \neq X_7 \): \( D_0 = \{2\} \implies \) filter \( 2 \notin D_7 \)
backtracking algorithm aka “branch-and-propagate”

1. **Propagation:**
   - for each constraint,
     - infer inconsistent value assignments
     - apply domain reduction
   - until fix point

2. **Tree Search:**
   - if domains are singleton, then solution found
   - if no domain is empty, then assign a free variable to a value
   - otherwise, backtrack
sudoku as a CSP with global constraints

\[ X_0, X_1, \ldots, X_{80} \]

\[ D_i = [0, 9] \quad \forall i \in [0, 80] \]

\[ X_0 = 2, \; X_1 = 6, \; \ldots \]

\text{alldifferent}(X_i)_{i \in l} \quad \forall l \in L

\text{alldifferent}(X_i)_{i \in c} \quad \forall c \in C

\text{alldifferent}(X_i)_{i \in s} \quad \forall s \in S
sudoku as a CSP with global constraints

\[ X_0, X_1, \ldots, X_{80} \]

\[ D_i = [0, 9] \quad \forall i \in [0, 80] \]

\[ X_0 = 2, \; X_1 = 6, \; \ldots \]

\textbf{alldifferent} \((X_i)_{i \in l} \quad \forall l \in L\)

\textbf{alldifferent} \((X_i)_{i \in c} \quad \forall c \in C\)

\textbf{alldifferent} \((X_i)_{i \in s} \quad \forall s \in S\)

\[ \text{global AC: } X_{43} \neq 7 \]
sudoku as a CSP with global constraints

\[ X_0, X_1, \ldots, X_{80} \]

\[ D_i = [0, 9] \quad \forall i \in [0, 80] \]

\[ X_0 = 2, \; X_1 = 6, \ldots \]

\[ \text{alldifferent}(X_i)_{i \in l} \quad \forall l \in L \]
\[ \text{alldifferent}(X_i)_{i \in c} \quad \forall c \in C \]
\[ \text{alldifferent}(X_i)_{i \in s} \quad \forall s \in S \]

\text{global AC: } X_{43} \neq 7

\text{alldifferent} \approx \text{bipartite matching } O(m \sqrt{n}) \; [\text{Régin 94}]
examples of value global constraints

- \texttt{alldifferent}((X_1, X_2, \ldots, X_n)) \ [\text{Régis 94}]
- \texttt{global-cardinality}((X_1, X_2, \ldots, X_n), (l_j)_j, (u_j)_j) \ [\text{Régis 96}]
- \texttt{among}(Z, (X_1, X_2, \ldots, X_n), V) \ [\text{Bessière et al. 05}]
- \texttt{soft-alldifferent}(Z, (X_1, X_2, \ldots, X_n)) \ [\text{Petit et al. 01}]
- \texttt{mincost-alldifferent}(Z, (X_1, X_2, \ldots, X_n), (c_{ij})_{i,j}) \ [\text{Sellmann 02}]

see also the Global Constraint Catalog \texttt{http://sofdem.github.io/gccat/}
examples of value global constraints

- \textbf{alldifferent}((X_1, X_2, ..., X_n)) [Régin 94]
- \textbf{global-cardinality}((X_1, X_2, ..., X_n), (l_j)_j, (u_j)_j) [Régin 96]
- \textbf{among}(Z, (X_1, X_2, ..., X_n), V) [Bessière et al. 05]
- \textbf{soft-alldifferent}(Z, (X_1, X_2, ..., X_n)) [Petit et al. 01]
- \textbf{mincost-alldifferent}(Z, (X_1, X_2, ..., X_n), (c_{ij})_{i,j}) [Sellmann 02]

see also the \textbf{Global Constraint Catalog} \url{http://sofdem.github.io/gccat/}

from consistency to filtering

- robustness and incrementality
- level of consistency vs. computation time
a CSP model for NSP

<table>
<thead>
<tr>
<th>employees</th>
<th>N</th>
<th>N</th>
<th>E</th>
<th>X_0^2</th>
<th>X_1^2</th>
<th>X_2^2</th>
<th>X_3^2</th>
<th>X_4^2</th>
<th>X_5^2</th>
<th>X_6^2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R</td>
<td>M</td>
<td>N</td>
<td>X_1^1</td>
<td>N</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>N</td>
<td>N</td>
<td>X_3^3</td>
<td>R</td>
<td>R</td>
<td>M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td>X_5^4</td>
<td>E</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>R</td>
<td>M</td>
<td></td>
<td></td>
<td></td>
<td>X_3^5</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

The diagram illustrates the relationships and constraints within the CSP model for NSP, with `R` and `C` denoting different sets or conditions.
a CSP model for NSP

global_cardinality (gcc)
a CSP model for NSP

global_cardinality (gcc)
language global constraints
flexible solution #1
CSPs as languages

- CSP solution \((x_1, x_2, \ldots, x_n) = \text{word } x_1x_2\ldots x_n \in D^*\)
- CSP model = language representation
- (un)satisfiability = emptiness
CSPs as languages

- CSP solution \((x_1, x_2, \ldots, x_n) = \text{word } x_1 x_2 \ldots x_n \in D^*\)
- CSP model = language representation
- (un)satisfiability = emptiness

\[
\text{language global constraint family}
\]

\[
\text{language}((X_1, X_2, \ldots, X_n), L) \equiv X_1 X_2 \ldots X_n \in L
\]
CSPs as languages

- CSP solution \((x_1, x_2, \ldots, x_n) = \text{word } x_1 x_2 \ldots x_n \in D^*\)
- CSP model = language representation
- (un)satisfiability = emptiness

### Language global constraint family

\[
\text{language}((X_1, X_2, \ldots, X_n), \mathcal{L}) \equiv X_1 X_2 \ldots X_n \in \mathcal{L}
\]

- **regular**\(((X_1, X_2, \ldots, X_n), \Pi)\) [Pesant 04]
- **cost-regular**\((Z, (X_1, X_2, \ldots, X_n), \Pi, c)\) [Demassey 05]
- **context-free**\(((X_1, X_2, \ldots, X_n), G)\) [Sellman 06, Quimper 06]
- **multicost-regular**\(((Z_1, Z_2, \ldots, Z_p), (X_1, X_2, \ldots, X_n), \Pi, c)\) [Menana 09]
language \(< X_1, \ldots, X_n, \mathcal{L}>\)

the satisfiability problem

is \(\mathcal{L} \cap (D_1 \times \cdots \times D_n)\) empty?

the consistency problem for \(v \in D_i\)

is \(\mathcal{L} \cap (D_1 \times \cdots \times D_{i-1} \times \{v\} \times D_{i+1} \times \cdots \times D_n)\) empty?
regular (\( < X_1, \ldots, X_n >, \Pi = (Q, D, \Delta, q_0, F) \))

\[
\begin{align*}
\mathcal{L}(\Pi) \cap (D_1 \times \cdots \times D_n) &
\end{align*}
\]

**Graph connectivity** [Pesant03]

**State-decomposition** [Beldiceanu04]

\[
\begin{cases}
S_i \in Q, & i = 1..n \\
(S_i, X_i, S_{i+1}) \in \Delta, & i = 1..n
\end{cases}
\]
regular \((< X_1, \ldots, X_n >, \Pi = (Q, D, \Delta, q_0, F))\)
regular \((< X_1, \ldots, X_n >, \Pi = (Q, D, \Delta, q_0, F))\)
regular \( (< X_1, \ldots, X_n >, \Pi = (Q, D, \Delta, q_0, F)) \)
regular \((<X_1, \ldots, X_n>, \Pi = (Q, D, \Delta, q_0, F))\)

graph connexit y

state-decomposition [Beldiceanu04]

\[
\begin{align*}
S_i & \in Q, \\
(S_i, X_i, S_{i+1}) & \in \Delta,
\end{align*}
\] 

\(i = 1..n\)

Flexible optimization Flexible Solutions
regular \((< X_1, \ldots, X_n >, \Pi = (Q, D, \Delta, q_0, F))\)

\[
\text{graph connectivity [Pesant03]}
\]

\[
\text{state-decomposition [Beldiceanu04]}
\]

\[
\left\{
\begin{aligned}
S_i & \in Q, \\
(S_i, X_i, S_{i+1}) & \in \Delta,
\end{aligned}
\right. 
\quad i = 1..n
\]
regular \((<X_1, \ldots, X_n>, \Pi = (Q, D, \Delta, q_0, F))\)
regular \(< X_1, \ldots, X_n >, \Pi = (Q, D, \Delta, q_0, F)\)

\[
L(\Pi) \cap (D_1 \times \cdots \times D_n)
\]

graph connectivity [Pesant03]

state-decomposition [Beldiceanu04]

\[
\begin{cases}
S_i \in Q, \quad i = 1..n \\
(S_i, X_i, S_{i+1}) \in \Delta, \quad i = 1..n
\end{cases}
\]

\[
O(|\Delta_n|) \text{ with } |\Delta_n| \ll n|\Delta|
\]
optimization variants

**cost-regular** $(Z, <X_1, \ldots, X_n>, \Pi, c)$

\[
eq X_1 X_2 \ldots X_n \in \mathcal{L}(\Pi) \land \sum_i c_i X_i = Z
\]

- shortest/longest path problem
- $O(|\Delta_n|)$ bound consistency on $Z$

Ilog Solver, Choco [Demassey, Pesant & Rousseau 05]
### Optimization Variants

<table>
<thead>
<tr>
<th>cost-regular $(Z, &lt;X_1, \ldots, X_n&gt;, \Pi, c)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1X_2 \ldots X_n \in \mathcal{L}(\Pi) \land \sum_i c_i X_i = Z$</td>
</tr>
<tr>
<td>- shortest/longest path problem</td>
</tr>
<tr>
<td>- $O(</td>
</tr>
</tbody>
</table>

Ilog Solver, Choco [Demassey, Pesant & Rousseau 05]

<table>
<thead>
<tr>
<th>multicost-regular $(&lt;Z^1, \ldots, Z^p&gt;, &lt;X_1, \ldots, X_n&gt;, \Pi, &lt;c^1, \ldots, c^p&gt;)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1X_2 \ldots X_n \in \mathcal{L}(\Pi) \land \sum_i c^k_i X_i = Z^k (\forall k)$</td>
</tr>
<tr>
<td>- resource-constrained SPP/LPP (NP-hard)</td>
</tr>
<tr>
<td>- lagrangian relaxation $O(K</td>
</tr>
</tbody>
</table>

Choco [Menana & Demassey 09]
benefit of aggregation (1)

<table>
<thead>
<tr>
<th></th>
<th>individual</th>
<th>aggregate</th>
<th>unfolded</th>
</tr>
</thead>
<tbody>
<tr>
<td>full-time #states</td>
<td>5,782</td>
<td>682</td>
<td>230</td>
</tr>
<tr>
<td>full-time #transitions</td>
<td>40,402</td>
<td>4,768</td>
<td>400</td>
</tr>
<tr>
<td>part-time #states</td>
<td>4,401</td>
<td>385</td>
<td>421</td>
</tr>
<tr>
<td>part-time #transitions</td>
<td>30,729</td>
<td>2,689</td>
<td>681</td>
</tr>
</tbody>
</table>

Size of the automata for the ASAP/GPost hard instance for full-time and part-time contracts, \( n = 28 \)
benefit of aggregation (2)

+ assignment costs to minimize
+ cardinality \((l, p, o)\) constraints
1 employee, 96 timeslots
number of working activities \((a, b, \ldots)\) between 1 and 50
10 instances each
default backtracking of Choco in 10 minutes
benefit of aggregation (2)

+ assignment costs to minimize
+ cardinality \((l, p, o)\) constraints
1 employee, 96 timeslots
number of working activities \((a, b, \ldots)\) between 1 and 50
10 instances each
default backtracking of Choco in 10 minutes

|\(|A|\) | multicost-regular | \(|\text{\&}\text{ cost-regular}\) | cost-regular \(\&\) gcc |
|---|---|---|---|
|   | proof | best | \#nodes | proof | best | \#nodes | proof | best | \#nodes |
| 1 | 0.0 | 0.0 | 41 | 1.2 | 1.0 | 3654 | 0.3 | 0.2 | 225 |
| 2 | 0.1 | 0.1 | 68 | 2.1 | 0.9 | 1563 | 0.6 | 0.3 | 393 |
| 4 | 0.2 | 0.1 | 67 | 13.9 | 8.8 | 6401 | 2.9 | 2.3 | 1199 |
| 8 | 0.3 | 0.2 | 52 | 101.7 | 49.8 | 19637 | 17.9 | 13.2 | 3597 |
|10 | 0.4 | 0.4 | 63 | 297.2 | 167.8 | 44530 | 50.0 | 47.7 | 7615 |
|15 | 0.8 | 0.7 | 63 | 50\% unsolved | 58.1 | 47.1 | 6233 |
|20 | 1.2 | 1.0 | 64 | 90\% unsolved | 58.1 | 44.0 | 4577 |
|30 | 1.8 | 1.5 | 62 | 90\% unsolved | 20\% unsolved |
|50 | 5.0 | 4.8 | 65 | 100\% unsolved | 60\% unsolved |

best = times (s) to find an optimum, proof = time (s) to prove optimality
ChocoETP = DFA + CP + LNS

flexible solution for NSP
a chief nurse-friendly solution?

1. high-level language to express rules
2. automated tool to model rules
3. automated tool to aggregate rules
4. automated tool to solve rules
a chief nurse-friendly solution?

1. high-level language to express rules
2. automated tool to model rules → WFA/regexp
3. automated tool to aggregate rules → WFA intersection
4. automated tool to solve rules → multicost-regular
a chief nurse-friendly solution?

1. high-level language to express rules
2. automated tool to model rules → WFA/regexp
3. automated tool to aggregate rules → WFA intersection
4. automated tool to solve rules → multicost-regular
5. automated tool to minimize penalties → CP + LNS
a chief nurse-friendly solution?

1. high-level language to express rules
2. automated tool to model rules \( \rightarrow \) WFA/regexp
3. automated tool to aggregate rules \( \rightarrow \) WFA intersection
4. automated tool to solve rules \( \rightarrow \) multicost-regular
5. automated tool to minimize penalties \( \rightarrow \) CP + LNS

**ChocoETP**

- CP-based Large Neighborhood Search solver
- pluggable parsers
- based on Choco and dk.brics Java libraries
- [https://github.com/sofdem/chocoETP](https://github.com/sofdem/chocoETP)
flexibility and effectiveness

### hard ASAP instances

|         | \(|I \times T|\) | [Métivier 09] | ChocoETP |
|---------|-----------------|---------------|-----------|
|         | cpu             | cpu           | nodes     | bk        |
| Azaïez  | 13×28           | 233           | 6.3       | 4006      | 5574      |
| Sintef  | 24×21           | -             | 1.4       | 165       | 53        |
| Millar-2S-1.1 | 8×12   | 1             | 0.5       | 29        | 0         |
| Millar-2S-1  | 8×12   | 1             | 0.3       | 25        | 0         |
| Ozkarahan | 14×7    | 1             | 0.2       | 24        | 5         |

### soft ASAP instances

|         | \(|I \times T|\) | [Métivier 09] | ChocoETP |
|---------|-----------------|---------------|-----------|
|         | opt             | penalty       | cpu       | penalty   | cpu       |
| GPost   | 8×28            | 5             | 8         | 234       | 5         | 75        |
| GPost-B | 8×28            | 3             | -         | -         | 3         | 3         |
| LLR     | 27×7            | 301           | 314       | 119       | 320       | 114       |
| Valouxis | 16×28       | 20            | 160       | 3780      | 20        | 4879      |
| ORTEC01 | 16×31           | 270           | -         | -         | 290       | 2920      |

Comparison with an ad-hoc LNS solver [Métivier09]
Conclusion
**Flexible optimization**

<table>
<thead>
<tr>
<th>modular solutions for recurring problems with mutable constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>■ key of flexibility: decomposed models</td>
</tr>
<tr>
<td>■ key of effectiveness: aggregated algorithms</td>
</tr>
</tbody>
</table>

**Conclusion**
Flexible optimization

Modular solutions for recurring problems with mutable constraints

- Key of flexibility: decomposed models
- Key of effectiveness: aggregated algorithms

⇒ Automated composition
**flexible optimization**

<table>
<thead>
<tr>
<th>modular solutions for recurring problems with mutable constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>■ key of flexibility: decomposed models</td>
</tr>
<tr>
<td>■ key of effectiveness: aggregated algorithms</td>
</tr>
</tbody>
</table>

⇒ automated composition ⇒ constraint learning
Flexible optimization

Modular solutions for recurring problems with mutable constraints

- Key of flexibility: decomposed models
- Key of effectiveness: aggregated algorithms

⇒ Automated composition ⇒ Constraint learning

Tools for flexibility

- Automata and graphs
- Global constraints and propagation
- Decomposition methods in linear programming (e.g. [Demassey06])
- Linearization (e.g. [Côté13])
Bibliography

Bibliography