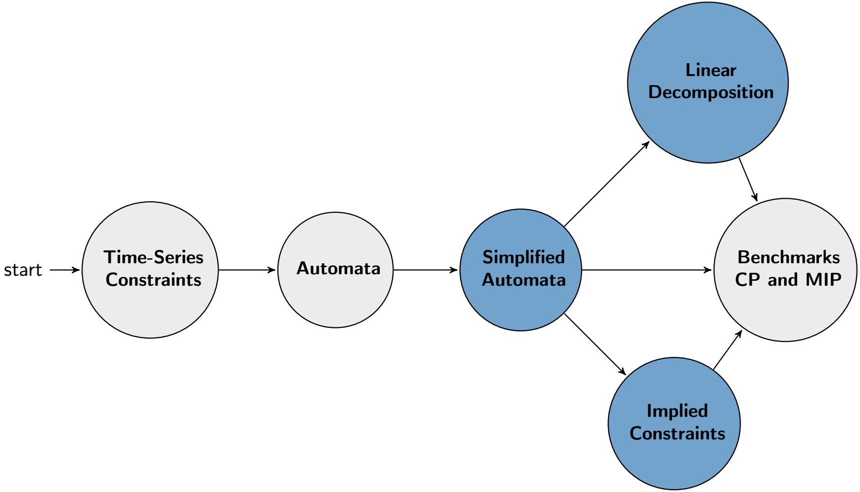
Time-Series Constraints: Improvements and Application in CP and MIP Contexts

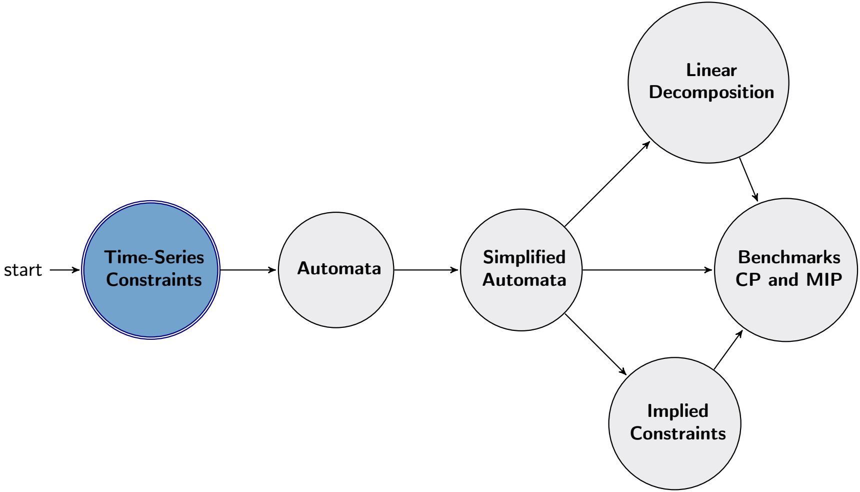
Ekaterina Arafailova, Nicolas Beldiceanu, Rémi Douence, Pierre Flener, M. Andreína Francisco R., Justin Pearson, and Helmut Simonis

May 31, 2016

Contents


start
Time-Series Constraints
Automata
Simplified Automata
Linear Decomposition
Implied Constraints
Benchmarks CP and MIP

Contents


start
Time-Series Constraints
Automata
Simplified Automata
Linear Decomposition
Benchmarks CP and MIP
Implied Constraints

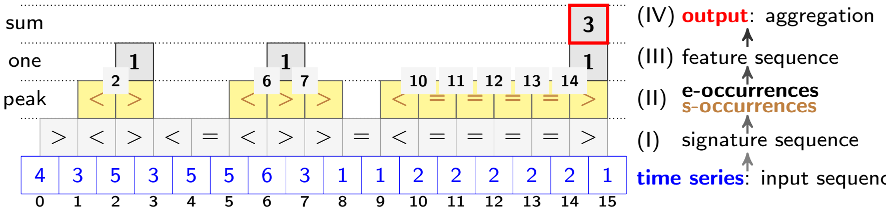
Time-series constraint

A time-series constraint[1] $g\_f\_\sigma(\langle X_1, \ldots, X_n \rangle, M)$ where every X_i is over $D_i \subset \mathbb{Z}$ is specified by

- A pattern, a regular expression over the alphabet $\{<, =, >\}$, e.g. Peak = '<(<|=)*(>|=)*>'.
  Currently 22 patterns in the framework
- A feature, a function over a subseries, e.g. one.
  Currently 5 features in the framework
- An aggregator, a function over a feature sequence, e.g. Sum.
  Currently 3 aggregators in the framework

[1] Beldiceanu, N., Carlsson, M., Douence, R., Simonis, H.: Using finite transducers for describing and synthesising structural time-series constraints. Constraints 21(1), 22-40 (January 2016): summary on p. 723 of LNCS 9255, Springer, 2015

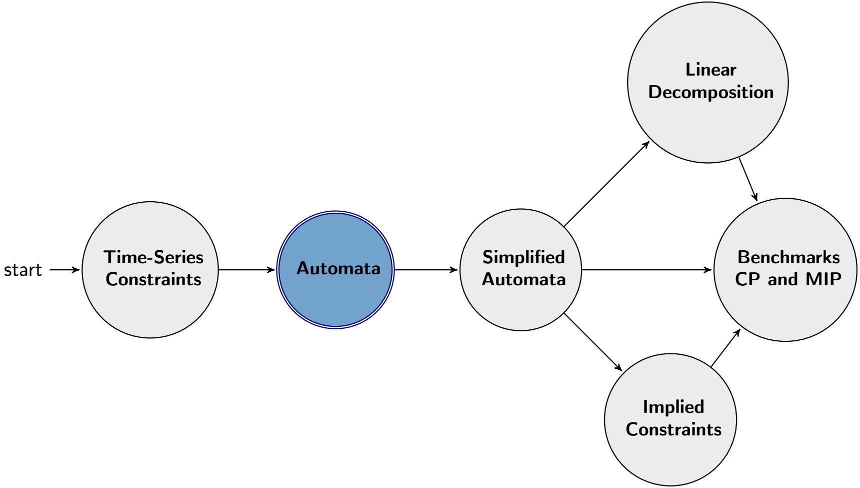
NbPeak



Example

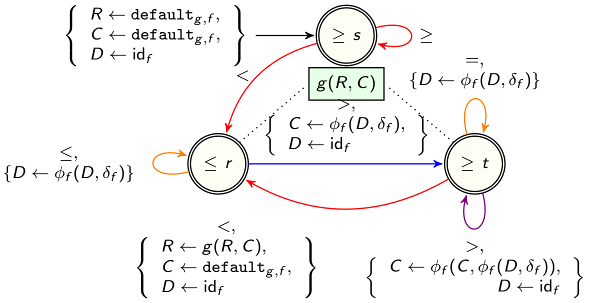
NbPeak$(\langle 4, 3, 5, 3, 5, 5, 6, 3, 1, 1, 2, 2, 2, 2, 2, 1\rangle, 3)$ holds !

Contents


start
Time-Series Constraints
Automata
Simplified Automata
Linear Decomposition
Implied Constraints
Benchmarks CP and MIP

Automata for time-series constraints

Every time-series constraint can be encoded as an automaton with three accumulators: $D(potential)$, $C(current)$, $R(aggregation)$



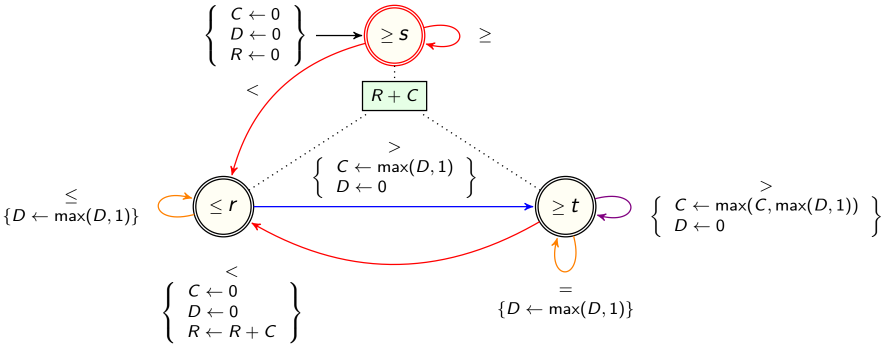
Automaton for the g_f_peak constraints.

Feature f	id_f	$\min_f$	$\max_f$	ϕ_f	δ_f^i
one	1	1	1	max	0

Aggregator g	$\mathrm{default}_{g,f}$
Sum	0

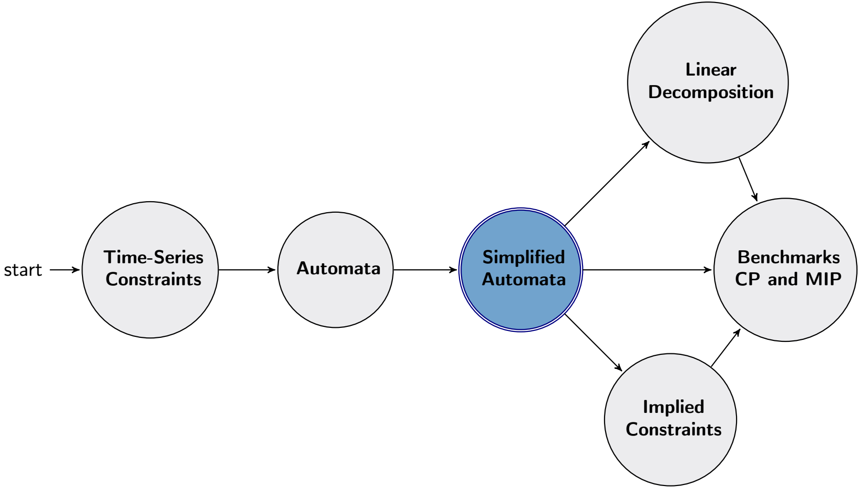
Automaton instantiation

When f is one and g is $\mathtt{Sum}$ the automaton becomes



Obviously, this automaton can be simplified

Contents


start
Time-Series Constraints
Automata
Simplified Automata
Linear Decomposition
Benchmarks CP and MIP
Implied Constraints

Automata simplifications

Goal

- **Reduce** the number of accumulators and aggregate as **early** as possible
- Simplify the automata at the stage of their synthesis

Three simplification types

- Simplifications coming from the properties of patterns, ex.: aggregate-once
- Simplifications coming from the properties of the feature/aggregator pairs, ex.: immediate-aggregation
- Removing the never used accumulators.

“Aggregate-once” simplification

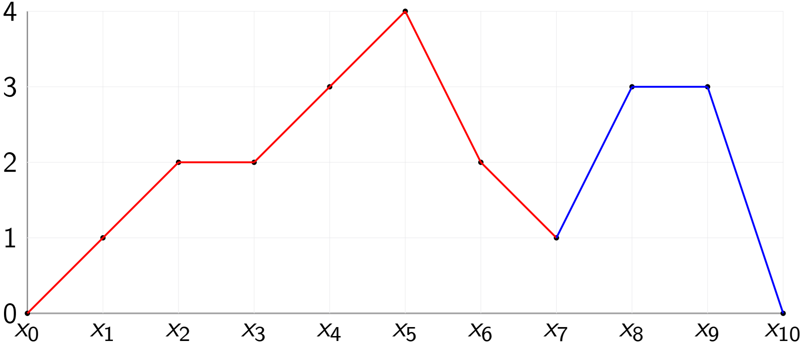
What is the “Aggregate-once” simplification ?

It allows to compute the feature value of a curent pattern occurrence only once and, possibly, earlier than the end of a pattern occurrence.

When is the simplification applicable ?

There must exist a transition on which the value of the feature from the current pattern occurrence is known.

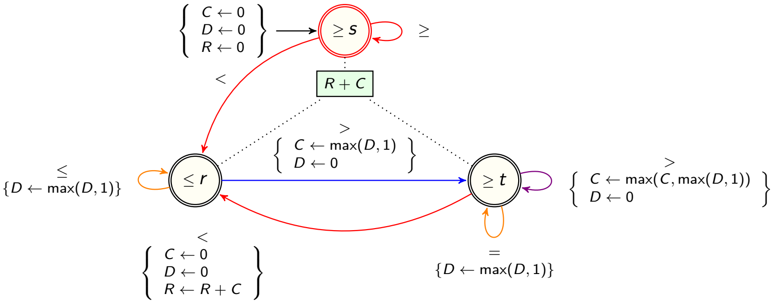
Example: counting number of peaks

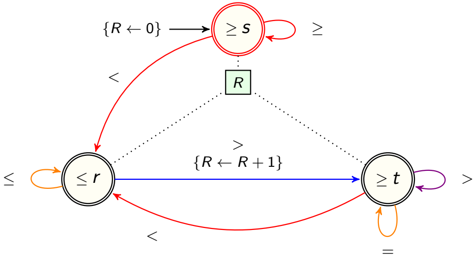


$s_0 =$‘<’ $s_1 =$‘<’ $s_2 =$‘=’ $s_3 =$‘<’ $s_4 =$‘<’ $s_5 =$‘>’ $s_6 =$‘>’ $s_7 =$‘<’ $s_8 =$‘=’ $s_9 =$‘>’

1. First peak is detected upon consuming s_5
2. Second peak is detected upon consuming s_9

Two automata for nb_peak

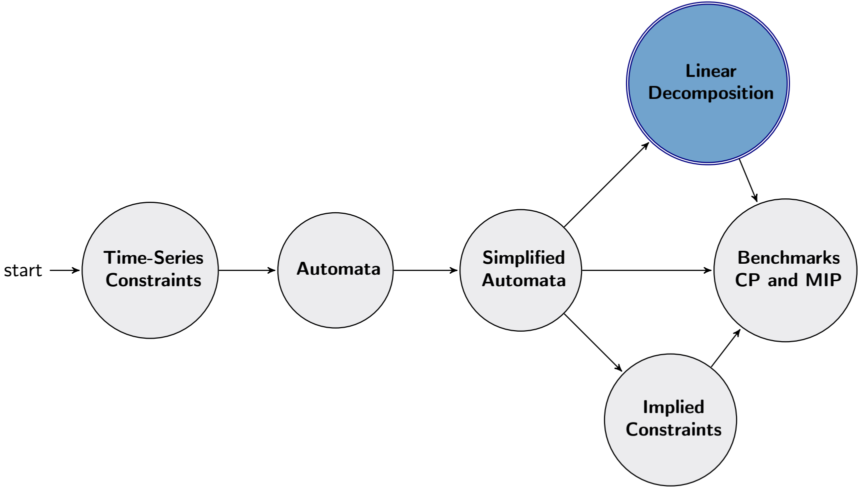




Percentage of simplified constraints

Simplification	Percentage
aggregate once	28.9 %
immediate aggreg.	45.9 %
other properties	11.6 %
unchanged automata	13.6 %

Contents



start
Time-Series Constraints
Automata
Simplified Automata
Linear Decomposition
Benchmarks CP and MIP
Implied Constraints

Input

Input

- Time-series variables X_i with i in $[0, n-1]$ over their domains $[a_i, b_i]$
- An automaton with accumulators for a time-series constraint with
  - a set of states Q;
  - an input alphabet Σ;
  - an m-tuple of integer accumulators with their initial values $I = \langle I_1, \ldots, I_m \rangle$;
  - a transition function $\delta : Q \times Z^m \times \Sigma \rightarrow Q \times Z^m$.

Goal

Goal

A way to generate a model for an automaton with linear or linearisable accumulator updates, for example containing min and max.

Linear decomposition of automata without accumulators

Côté, M.C., Gendron, B., Rousseau, L.M.: Modeling the regular constraint with integer programming. In: CPAIOR 2007. LNCS, vol. 4510, pp. 29–43. Springer (2007)

Signature constraint

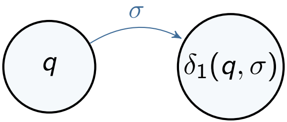
Introduced variables: S_i over Σ with $i \in [0, n-2]$.

What do the values of S_i mean ?

$$S_i = \text{`}{>}\text{'} \Leftrightarrow X_i > X_{i+1}, \forall i \in [0, n-2]$$
$$S_i = \text{`}{=}\text{'} \Leftrightarrow X_i = X_{i+1}, \forall i \in [0, n-2]$$
$$S_i = \text{`}{<}\text{'} \Leftrightarrow X_i < X_{i+1}, \forall i \in [0, n-2]$$

Transition function constraints

Introduced variables: Q_i over Q with $i \in [0, n-1]$; T_i over $Q \times \Sigma$ with $i \in [0, n-2]$



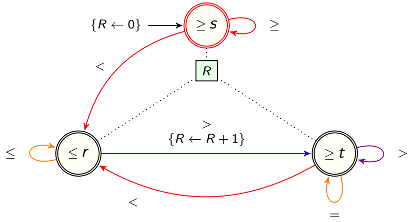
Each transition constraint has a form:

$Q_i = q \wedge S_i = \sigma \Leftrightarrow Q_{i+1} = \delta_1(q, \sigma) \wedge T_i = \langle q, \sigma \rangle$,
$\forall i \in [0, n-2],\ \forall q \in Q,\ \forall \sigma \in \Sigma$

Initial state is fixed

$Q_0 = q_0$

Accumulator updates



Accumulator updates

R_i over $[a, b]$ with i in $[0, n-1]$; T_i over $Q \times \Sigma$ with i in $[0, n-2]$.

- $R_0 = 0$
- $T_i = \langle r, > \rangle \Rightarrow \mathbf{R_{i+1} = R_i + 1}, \forall i \in [0, n-2]$
- $T_i = \langle q, \sigma \rangle \Rightarrow \mathbf{R_{i+1} = R_i}, \forall i \in [0, n-2], \forall \langle q, \sigma \rangle \in (Q \times \Sigma) \setminus \langle r, > \rangle$
- $M = R_{n-1}$

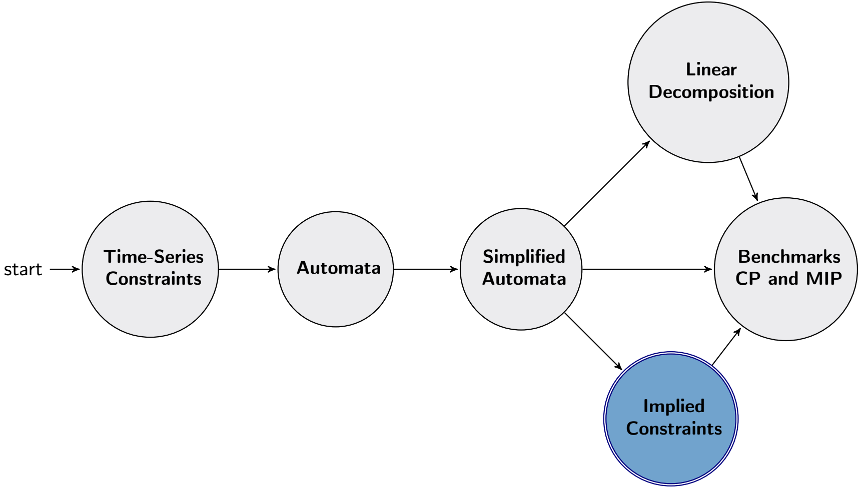
New variables for the linear model

New variables

- Q_i is replaced by 0-1 variables Q_i^q for all q in Q.
  $Q_i^q = 1 \Leftrightarrow Q_i = q$
- New constraint: $\sum_{q \in Q} Q_i^q = 1, \forall i \in [0, \ldots, n-1]$
- The same procedure for T_i and S_i wrt their domains
- X_i and R_i remain integer variables!

- Every constraint of the logical model is made linear by applying some standard techniques
- The linear model has $O(n)$ variables and $O(n)$ constraints

Contents


start
Time-Series Constraints
Automata
Simplified Automata
Linear Decomposition
Benchmarks CP and MIP
Implied Constraints

Implied constraints

Implied constraints[2] improves propagation for constraints encoded via automata with at least one accumulator

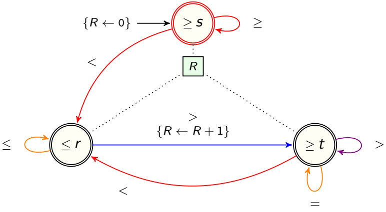
- The implied constraints are generated offline
- The implied constraints are of the form:

$$\alpha_1 y_1 + \cdots + \alpha_k y_k + \beta \geq 0$$

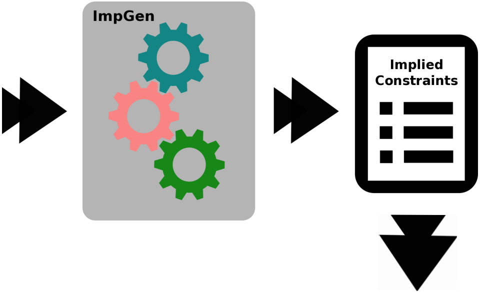
where the y_i are the accumulators of $\mathcal{A}(C,\ D,\ R)$ and the weights α_i and β are to be found

- Theoretically supported by Farkas' Lemma

[2] Francisco Rodríguez, M.A., Flener, P., Pearson, J.: Implied constraints for automaton constraints. In: GCAI 2015. EasyChair Epic Series in Computing, vol. 36, pp. 113–126. EasyChair (2015)



Options



$$R_{i+2} \leq R_i + 1$$

Improvements

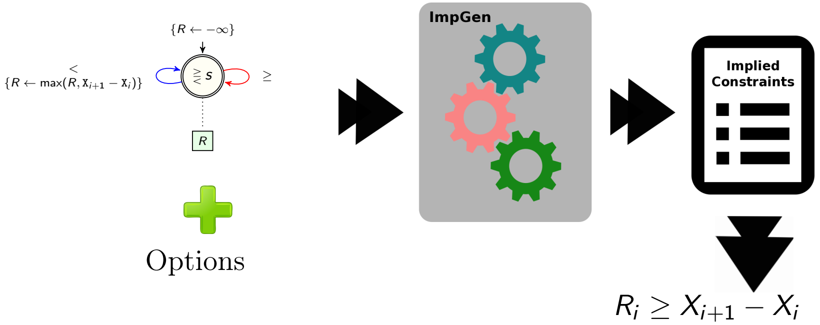
The first version of `ImpGen`

- Only linear accumulator updates
- Manual selection

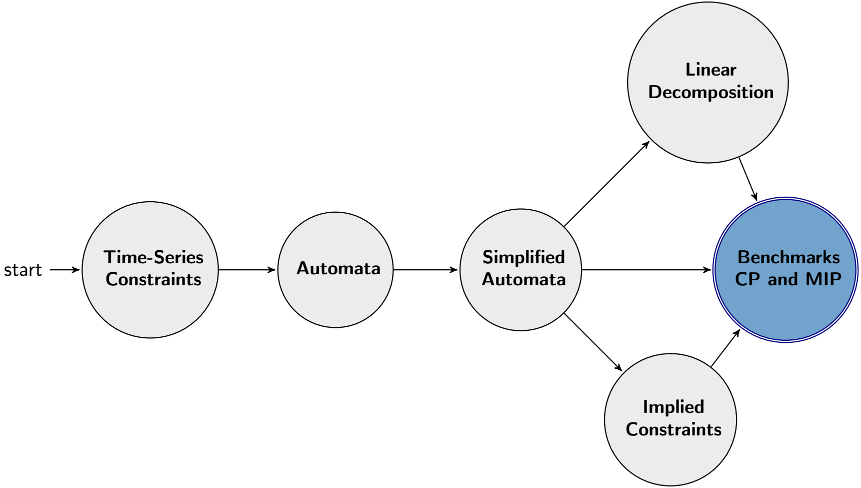
Improvements of the new version

- Can handle max and min in accumulators updates
- Automatic selection by ranking

Improvements of `ImpGen`: example



Contents


start
Time-Series Constraints
Automata
Simplified Automata
Linear Decomposition
Implied Constraints
Benchmarks CP and MIP

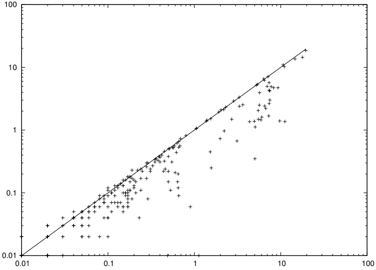
Benchmark CP

Goal

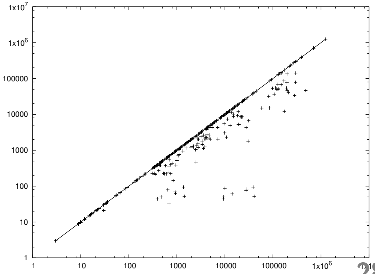
compare original and simplified automata

- For every time-series constraint maximise the result
- Time series of length 15 over $[1, 3]$
- Timeout of 100 seconds

(a) Runtime

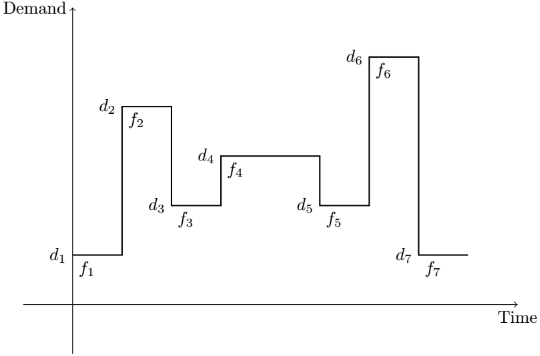


(b) Backtracks



Staff scheduling application

Figure 1: Typical Benchmark Demand Profile



- Satisfy the demand;
- Take into account business rules
- Respect union's rules
- Minimise the costs

Results for staff scheduling application

- P characterises complexity of the problem
- Consider $P \in \{10, 15, 20, 25, 30, 35, 40\}$
- 100 instances for every value of P

optimality gap					
	cp		mip		
p	avg	max	avg	max	opt
20	3.42	9.67	2.28	18.77	27/100
30	3.20	8.02	2.04	6.34	26/100
40	3.51	17.32	1.97	10.47	18/100

- In average MIP is always better
- The maximal gap sometimes is smaller for CP
- MIP can solve to optimality just few instances

Conclusion

Contributions of the paper

- A linear decomposition for time-series constraints with $O(n)$ variables and $O(n)$ constraints
- Simplified automata for time-series constraints
- New version of the generator of linear implied constraints which handles accumulator updates with min, max
- Benchmarks in the contexts of CP and MIP

Thank you for your attention!
Questions?