



MiniBrass: Soft Constraint Programming

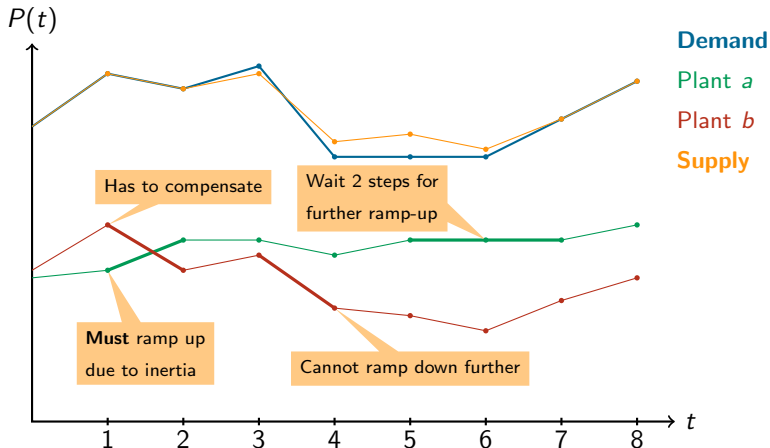
Alexander Schiendorfer et al.



Wo wir letztes Jahr waren . . .

Fahrplanerstellungproblem

Ziel: Plane Kraftwerke so ein, dass sie die Last gemeinsam erfüllen

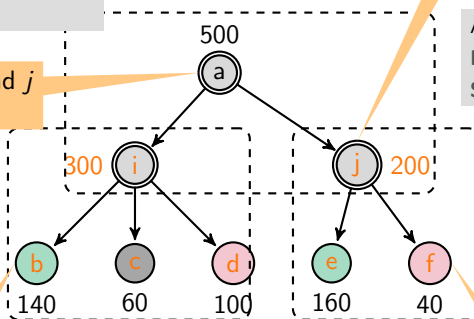


Regio-zentrale Fahrpläne
ICAART'14, SAOS'14
Marktbasiert
TAAS'15

Wie kann ich *e* und *f* repräsentieren?

Abstraktion
ICAART'14, TCCI'15
SASO'15

Was sollen *i* und *j*
beisteuern?



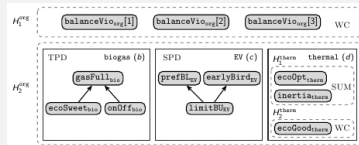
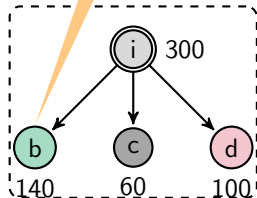
Supply Automata
SEN-MAS'14
TCCI'15

Wie vermeide ich
meinen Speicher
über 90% zu füllen?

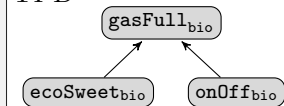
Constraint Relationships / PVS
SGAI'13, ICTAI'14
Wirsing'15, Constraints'16

Wie beschreibe ich
meine Abläufe?

Wie vermeide ich
meinen Speicher
über 90% zu füllen?



TPD



Ziel: Integration von Individualpräferenzen.

Constraint-Problem $((X, D), C)$

- Variablen X , Domänen $D = (D_x)_{x \in X}$, Constraints C

In der Praxis: unerfüllbare Probleme

$((\{x, y, z\}, D_x = D_y = D_z = \{1, 2, 3\}), \{c_1, c_2, c_3\})$ mit

$$c_1 : x + 1 = y$$

$$c_2 : z = y + 2$$

$$c_3 : x + y \leq 3$$

- Nicht alle Constraints können gleichzeitig erfüllt werden
 - z. B., c_2 erzwingt $z = 3$ und $y = 1$, im Konflikt zu c_1
- Ein Agent wählt also zwischen Belegungen, die $\{c_1, c_3\}$ oder $\{c_2, c_3\}$ erfüllen.

Welche Belegungen $v \in [X \rightarrow D]$ sollen bevorzugt werden?

Harte Constraints aus Supply Automata:

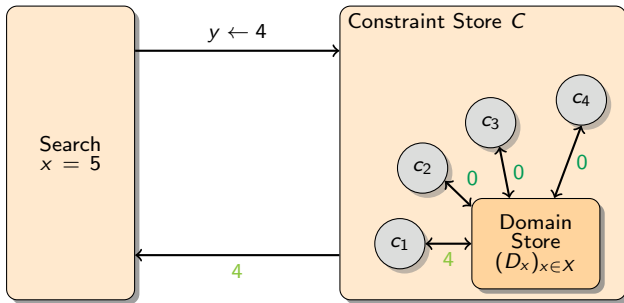
$$\text{hardBounds} : \forall t \in T, a \in A : m[a][t] = \text{on} \rightarrow P_{\min} \leq S[a][t] \leq P_{\max}$$

Weiche Constraints anlagenspezifisch (z.B. Präferenz für 350 bis 390 KW):

$$\text{ecoSweet}_{\text{bio}} : \forall t \in T : m[\text{biogas}][t] = \text{on} \rightarrow 350 \leq S[\text{biogas}][t] \leq 390$$

oder Änderungsgeschwindigkeit

$$\text{inertia}_{\text{therm}} : \forall t \in T : |S[\text{biogas}][t] - S[\text{biogas}][t + 1]| \leq 10$$



- Eine Menge von Bewertungen, z.B., $\{0, \dots, k\}$
- Eine Kombination +
- Ein neutrales Element 0
- Eine partielle Ordnung (\mathbb{N}, \geq) mit 0 als Top

Genannt **valuation structure** (?), bei totaler Ordnung, ansonsten **partial valuation structure** (?). Ähnlich: (?): c-Semiringe

Zugrundeliegende **algebraische Struktur**: Partielle Bewertungsstruktur (*partial valuation structure, partiell geordnetes, kommutatives Monoid*)

- $(M, \cdot_M, \varepsilon_M, \leq_M)$
- $m \cdot_m \varepsilon_M = m$
- $m \leq_M \varepsilon_M$
- $m \leq_M n \rightarrow m \cdot_M o \leq_M n \cdot_M o$

Abstrakt

- M ... Elemente
- \cdot_M ... Kombination von Bewertungen
- ε_M ... neutrales, "bestes" Element
- \leq_M ... Ordnung, links "schlechter"

Konkret

- $\{0, \dots, k\}$
- $+_k$
- 0
- \geq

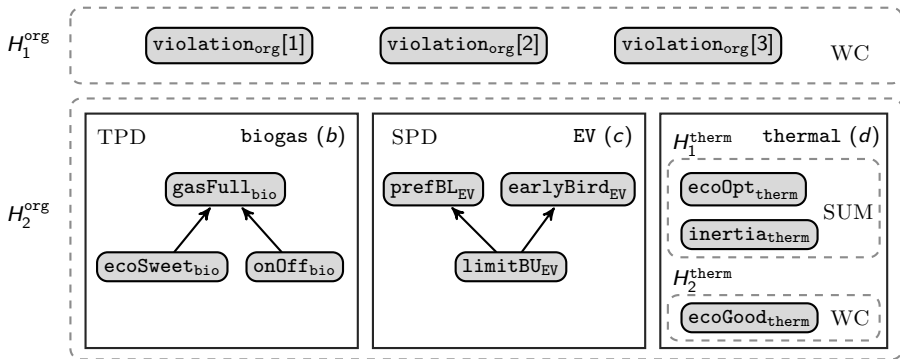
(?; ?)

Konkrete PVS	M	\oplus_M	\leq_M	ε_M
Weighted CSP (WCSP)	\mathbb{N}	$+$	\geq	0
Fuzzy CSP	$[0, 1]$	\min	\leq	1
Inclusion Max CSP	2^{C_s}	\cup	\supseteq	\emptyset
Constraint Relationships ¹	$\mathcal{M}^{\text{fin}}(C_s)$	\cup	\supseteq_{SPD}	\emptyset

Hauptidee

Implementiere Lösungsverfahren für Constraint-Probleme, die durch Bewertungsstrukturen geordnet sind. Instantiiere für konkrete Probleme.

¹ C_s is the set of soft constraints, \supseteq_{SPD} is the SPD-ordering on sets.



Die Präferenzstruktur dieses Problems:

$$V_{org_1} \times (P_{biogas} \times P_{EV} \times (P_{thermal}^1 \times P_{thermal}^2))$$

SPD ... Single-Pred.-Dom.

TPD ... Single-Pred.-Dom.

SUM ... Sum of Errors

WC ... Worst-Case Error

Third International CSP Competition (CPAI'08)

Max-CSP and WCSP solvers submitted

Max-CSP: 4 submitted solvers (and a few more versions)

AbsconMax a CSP solver in Java

CSP4J a CSP library in Java

Sugar a SAT based solver

toulbar2 a WCSP solver

WCSP: only one solver submitted (Toulbar)

- the competition has been postponed

Im Constraint Programming:

- Fokus auf *klassischen* Constraint-Lösern
- Erweiterung auf einfache Optimierung (Branch & Bound)
- Zielfunktion kann skalare Variable (`int` oder `float`) sein
- **toulbar2** ist der einzige dedizierte Weighted-CSP-Solver

In der mathematischen Programmierung:

- Probleme müssen gewisse Struktur aufweisen (lineare Constraints, quadratische Constraints, etc.)
- Schlecht geeignet für beliebige Ordnungen nach denen optimiert werden soll

Wir wollen aber heterogene PVS → **MiniZinc**


Rationale

Eine Modellierungssprache – viele Solver

Reduziere Soft-Constraint-Probleme auf konventionelle Constraint-Probleme



- Gecode (CP)
- JaCoP (CP)
- Google Optimization Tools (CP)
- CPLEX (CP/LP/MIP)
- G12 (CP/LP/MIP)
- ...




[View on GitHub](#) 

MiniBrass*

A utility library for soft constraints with constraint relationships on top of the MiniZinc/MiniSearch toolchain.

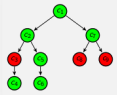
 tar.gz  .zip



Optimization with Preferences

Many combinatorial optimization problems are conveniently expressed using a constraint-based modeling language. They are then solved by powerful constraint programming or mathematical programming solvers.

We provide support for over-constrained problems or problems where desirable properties can be modeled as optional (soft) constraints. Importance is expressed only by



<http://isse-augsburg.github.io/minibrass/>

Basismodell (MiniZinc)

```
include "hello_o.mzn";
include "soft_constraints/
    pvs_gen_search.mzn";
% the basic, "classic" CSP
set of int: DOM = 1..3;

var DOM: x; var DOM: y;
var DOM: z;
% add. *hard* constraints
% e.g. constraint x < y;

solve search pvs_BAB();
```

Solution: x = 1; y = 2; z = 1

Valuations: mbr_overall_cr1 = 2..2

=====

Präferenzmodell (MiniBrass)

```
PVS: cr1 =
  new ConstraintRelationships("cr1") {
    soft-constraint c1: 'x + 1 = y';
    soft-constraint c2: 'z = y + 2';
    soft-constraint c3: 'x + y <= 3';

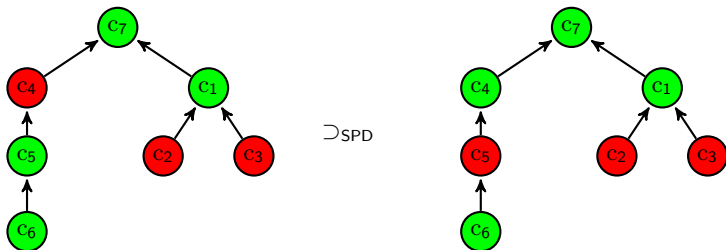
    crEdges : '[| mbr.c2, mbr.c1 |
                mbr.c3, mbr.c1 |]';
    useSPD: 'false' ;
  };

solve cr1;
```


isWorseThan-Relation für Mengen verletztter Constraints (?)

$$V \uplus \{c\} \supset_{\text{SPD}} V$$

$$V \uplus \{c_{\text{imp}}\} \supset_{\text{SPD}} V \uplus \{c_{\neg\text{imp}}\} \quad \text{if } c_{\neg\text{imp}} \rightarrow c_{\text{imp}}$$

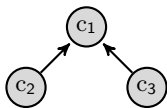


- Bekannt als *Smyth-Ordnung* (Powerdomains)
- Entsteht aus *freier Konstruktion* über Constraint-Relationship.

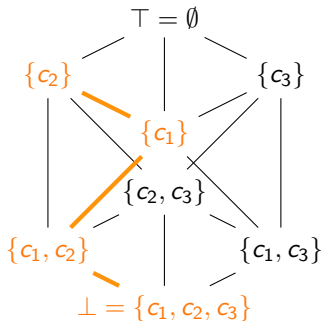
(?, Ch. 9)

(?)

Der partiell geordnete Bewertungsraum



c1: 'x + 1 = y';
c2: 'z = y + 2';
c3: 'x + y <= 3';



x = 1; y = 1; z = 1

Valuation = 1..2

x = 1; y = 1; z = 3

Valuation = 1..1

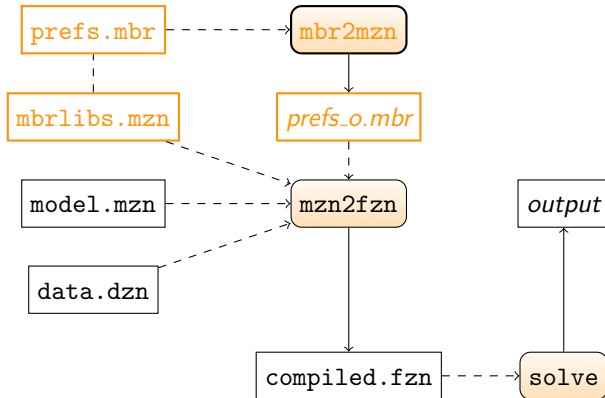
x = 1; y = 2; z = 1

Valuations = 2..2

=====

```
function ann: pvs_BAB() =  
  repeat(  
    if next() then  
      print("Intermediate solution:") /\ print() /\  
      commit() /\ postGetBetter()  
    else break endif    );
```

Architektur



Konzepte, Sprachdesign MiniBrass

- AS, Alexander Knapp, Gerrit Anders, Oliver Kosak

Anwendungen, Multiagenten-Einsatz

- Alexander Schubert (MSc-Thesis: Einsatz von Voting-Verfahren), Markus Tolls (MSc-Thesis: Formalisierung von Task-Allocation-Problemen)

- Vortrag Helmholtz-Zentrum München
- Vortrag FH Hagenberg
- Tutorial @ SASO 2016

