Constraint Relationships for Soft Constraints

A. Schiendorfer, J.P. Steghöfer, A. Knapp, F. Nafz, W. Reif

Universität
Augsburg
University

Motivation

Institute for Software & Systems Engineering

Constraint programming

- Declarative paradigm to problem solving
- Specifying problems as *variables*, *domains*, and *constraints*
- A *versatile* tool for combinatorial and optimisation problems (resource allocation, job-shop scheduling etc)

Motivation

Institute for Software & Systems Engineering

Constraint programming

- Declarative paradigm to problem solving
- Specifying problems as *variables*, *domains*, and *constraints*
- A *versatile* tool for combinatorial and optimisation problems (resource allocation, job-shop scheduling etc)

In practise:

- Problems tend to be over-constrained
- Difficult to distinguish betw. "real constraints" and *preferences*
- $\rightarrow$ Preference formalisms in AI (soft constraints, CP-nets, constraint hierarchies, . . .)

Existing solutions I

Institute for Software & Systems Engineering

Soft Constraints

- Some constraints are *hard*, some are *soft*
- Different kinds: Weighted CSP, Fuzzy CSP etc.
- *Quantify* satisfaction (or violation) degree of a solution
- Generalized as c-semiring or valued constraints

CP-nets

- Capture preferences over domains with respect to other variables' assignments
- Specify total orders for all possible assignments of the parent variables
- Requires considerable specification amount and knowledge about structure and assignments

Existing solutions II

Institute for Software & Systems Engineering

Constraint Hierarchies

- Categorize constraints on hierarchy levels
- Start with most important constraints, then follow-up levels as well as possible
- Most closely related to our approach

Problems

Institute for Software & Systems Engineering

Quantitative Formalisms (Soft Constraints etc.)

- Hard to capture
- Problems with normalizations with different scales
- Rationales/Motivations can get lost

Problems

Institute for Software & Systems Engineering

Quantitative Formalisms (Soft Constraints etc.)

- Hard to capture
- Problems with normalizations with different scales
- Rationales/Motivations can get lost

CP-Nets

- Specification overhead with total orders for *domain values* (worst case $D_1 \times \cdots \times D_n$ total orders to be specified)
- Not suited in situations where preference is defined over constraints rather than values

Problems

Institute for Software & Systems Engineering

Quantitative Formalisms (Soft Constraints etc.)

- Hard to capture
- Problems with normalizations with different scales
- Rationales/Motivations can get lost

CP-Nets

- Specification overhead with total orders for *domain values* (worst case $D_1 \times \cdots \times D_n$ total orders to be specified)
- Not suited in situations where preference is defined over constraints rather than values

Constraint Hierarchies (CH)

- Strong hierarchical notation (constraint on higher level is strongly more important)
- Difficult when combining different hierarchies
- We generalise one subclass of CH

Constraint Relationships

Strategy

- Define a partial order over constraints
- Lift that order to violation sets of solutions (*dominance property*)
- Calculate weights for constraints based on CR

Constraint Relationships

Strategy

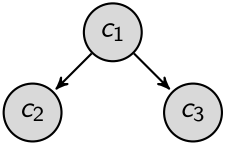
- Define a partial order over constraints
- Lift that order to violation sets of solutions (*dominance property*)
- Calculate weights for constraints based on CR

Benefits

- Qualitative formalism – keep underlying rationales!
- Combine several sets of constraint relationships in MAS
- Lightweight formalism – easy to specify
- Transformation into weighted CSP – better understood
- Coarser granularity than CP-nets

Constraint Relationships – An example

- Variables: X, Y, Z
- Domains: $\{0, 1, 2\}$
- Constraints:
  - $c_1 : x + 1 = y$
  - $c_2 : z = y + 2$
  - $c_3 : x + y \leq 3$



- Not all three constraints can be satisfied simultaneously
- E.g. c_2 forces z to be 2 and y to be 0, conflicting with c_1
- We can choose between solutions satisfying $\{c_1, c_3\}$ or $\{c_2, c_3\}$
- How to settle this conflict?

Semantics of Dominance Properties

- CR specify which constraint is more important than another one
- We need to tell *how* much more important a single constraint is!
- T, U stand for the set of constraints assignments t and v *violate*
- p denotes the used dominance property, $\uplus$ is disjoint union, R is the set of CR
- Read $\longrightarrow_R^p$ as "worsens to"
- $t >_R^p u \leftrightarrow T \; (\longrightarrow_R^p)^+ \; U$

Semantics of Dominance Properties

Institute for Software & Systems Engineering

- CR specify which constraint is more important than another one
- We need to tell *how* much more important a single constraint is!
- T,U stand for the set of constraints assignments t and v *violate*
- p denotes the used dominance property, $\uplus$ is disjoint union, R is the set of CR
- Read $\longrightarrow_R^p$ as "worsens to"
- $t >_R^p u \leftrightarrow T\ (\longrightarrow_R^p)^+\ U$

$$T \longrightarrow_R^p T \uplus \{c\} \tag{W1}$$

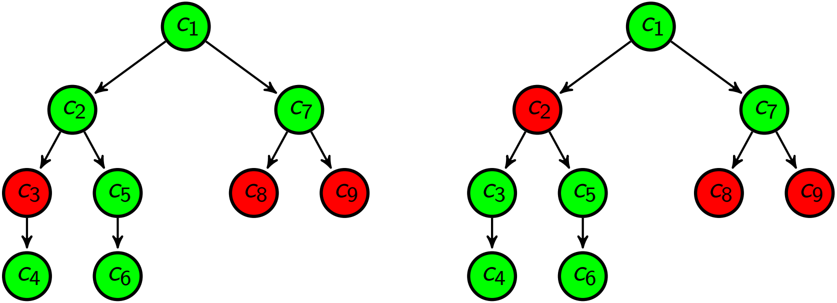
$$\frac{T_1 \longrightarrow_R^p U_1 \quad T_2 \longrightarrow_R^p U_2}{T_1 \uplus T_2 \longrightarrow_R^p U_1 \uplus U_2} \tag{W2}$$

Single-Predecessor-Dominance

Institute for
Software & Systems
Engineering

$$T \uplus \{c\} \longrightarrow_R^{\mathrm{SPD}} T \uplus \{c'\} \quad \text{if } c \prec_R c' \tag{SPD}$$

SPD example



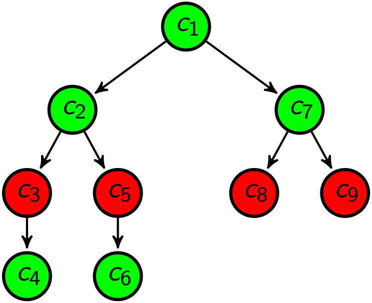
Direct-Predecessor-Dominance

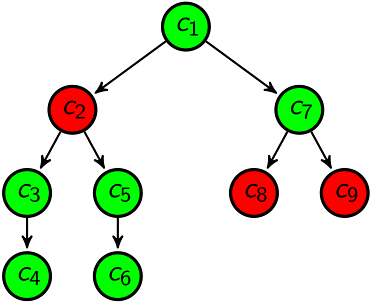
Institute for Software & Systems Engineering

$$T \uplus \{c_1, \ldots, c_k\} \longrightarrow_R^{\mathrm{DPD}} T \uplus \{c'\} \qquad \text{(DPD)}$$

$$\text{if } \forall c \in \{c_1, \ldots, c_k\} : c \prec_R c'$$

DPD example





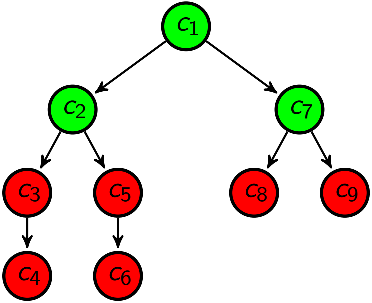
Transitive-Predecessor-Dominance

Institute for Software & Systems Engineering

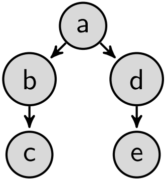
$$T \uplus \{c_1, \ldots, c_k\} \longrightarrow_R^{\mathrm{TPD}} T \uplus \{c'\} \qquad \text{(TPD)}$$

$$\text{if } \forall c \in \{c_1, \ldots, c_k\} : c \prec_{R^+} c'$$

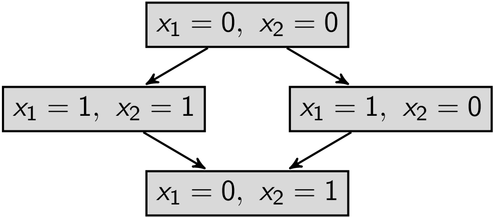
TPD example



CR compared to other formalisms



A set of constraint relationships not expressible in LPB-hierarchies



Desired solution order, $x_1 = 0,\ x_2 = 0$ should be the best.

Weight transformation

Institute for Software & Systems Engineering

For practical purposes, we can transform the solution order induced by constraint relationships into a numerical mapping reflecting that order.

$$w_R^{\mathrm{SPD}}(c) = 1 + \max\{w_R^{\mathrm{SPD}}(c') \mid c' \in \mathcal{C} : c \succ_R c'\} \quad \text{for } c \in \mathcal{C} \ .$$

Weight transformation

Institute for Software & Systems Engineering

For practical purposes, we can transform the solution order induced by constraint relationships into a numerical mapping reflecting that order.

$$w_R^{\mathrm{SPD}}(c) = 1 + \max\{w_R^{\mathrm{SPD}}(c') \mid c' \in \mathcal{C} : c \succ_R c'\} \quad \text{for } c \in \mathcal{C} \ .$$

$$w_R^{\mathrm{DPD}}(c) = 1 + \sum_{c' \in \mathcal{C} : c \succ_R c'} w_R^{\mathrm{DPD}}(c') \quad \text{for } c \in \mathcal{C} \ .$$

Weight transformation

Institute for Software & Systems Engineering

For practical purposes, we can transform the solution order induced by constraint relationships into a numerical mapping reflecting that order.

$$w_R^{\mathrm{SPD}}(c) = 1 + \max\{w_R^{\mathrm{SPD}}(c') \mid c' \in \mathcal{C} : c \succ_R c'\} \quad \text{for } c \in \mathcal{C} \ .$$

$$w_R^{\mathrm{DPD}}(c) = 1 + \sum_{c' \in \mathcal{C} : c \succ_R c'} w_R^{\mathrm{DPD}}(c') \quad \text{for } c \in \mathcal{C} \ .$$

$$w_R^{\mathrm{TPD}}(c) = 1 + \sum_{c' \in \mathcal{C} : c \succ_R c'} (2 \cdot w_R^{\mathrm{TPD}}(c') - 1) \quad \text{for } c \in \mathcal{C} \ .$$

Weight transformation

Institute for
Software & Systems
Engineering

For practical purposes, we can transform the solution order induced by constraint relationships into a numerical mapping reflecting that order.

$$w_R^{\mathrm{SPD}}(c) = 1 + \max\{w_R^{\mathrm{SPD}}(c') \mid c' \in \mathcal{C} : c \succ_R c'\} \quad \text{for } c \in \mathcal{C} \;.$$

$$w_R^{\mathrm{DPD}}(c) = 1 + \sum_{c' \in \mathcal{C} : c \succ_R c'} w_R^{\mathrm{DPD}}(c') \quad \text{for } c \in \mathcal{C} \;.$$

$$w_R^{\mathrm{TPD}}(c) = 1 + \sum_{c' \in \mathcal{C} : c \succ_R c'} (2 \cdot w_R^{\mathrm{TPD}}(c') - 1) \quad \text{for } c \in \mathcal{C} \;.$$

If $t >_R^p u$, then $W_R^p(T) < W_R^p(U)$ for $p \in \{\mathrm{SPD}, \mathrm{DPD}, \mathrm{TPD}\}$.

Embedding in constraint optimisation

Institute for Software & Systems Engineering

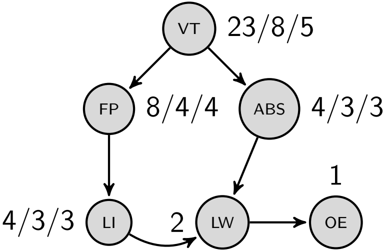
Each constraint $c \in \mathcal{C}$ is reformulated as follows (p_c denotes a penalty vector indexed by the set of constraints):

$$c' : (c \wedge p_c = 0) \vee (\neg c \wedge p_c = w(c))$$

where. This allows a constraint optimizer to either fulfill c or set $p_c \neq 0 \in \mathbb{N}$, the penalty for violating c, to the weight of the corresponding constraint.

$$\min \sum_{c \in \mathcal{C}} p_c, s.t. \sum_{c \in \mathcal{C}} p_c < k$$

Constraint Relationships – Ski-Day-Planner



- Avoid black slopes (ABS)
- Variety (VT)
- Fun-park (FP)
- Little Wait (LW)
- Only Easy Slopes (OE)
- Lunch Included (LI)

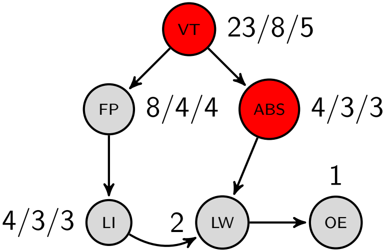
$$t_X^{(1)} \not\models \{\mathrm{VT}, \mathrm{ABS}\}$$

$$t_X^{(2)} \not\models \{\mathrm{FP}, \mathrm{ABS}, \mathrm{LW}, \mathrm{LI}, \mathrm{OE}\}$$

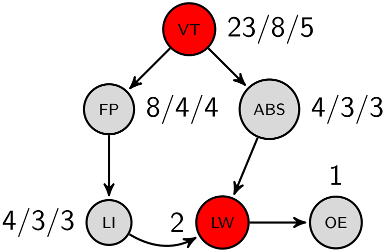
$$t_X^{(3)} \not\models \{\mathrm{VT}, \mathrm{LW}\}$$

Constraint Relationships – Ski-Day-Planner

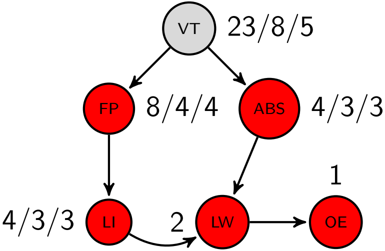
Weights are given for SPD/DPD/TPD



$t_X^{(1)} \not\models \{\mathrm{VT}, \mathrm{ABS}\}$
Penalty: 27/11/8

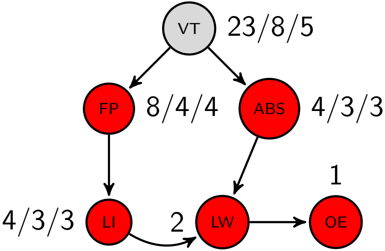


$t_X^{(3)} \not\models \{\mathrm{VT}, \mathrm{LW}\}$
Penalty: 25/**10**/**7**



$t_X^{(2)} \not\models \{\mathrm{FP}, \mathrm{ABS}, \mathrm{LW}, \mathrm{LI}, \mathrm{OE}\}$

Penalty: **19**/13/13



$t_X^{(2)} \not\models \{\mathsf{FP}, \mathsf{ABS}, \mathsf{LW}, \mathsf{LI}, \mathsf{OE}\}$

Penalty: **19**/13/13

→ this is what Constraint Hierarchies would do

Future work

Institute for Software & Systems Engineering

Theoretical

- C-semiring solely based on constraint relationships
- Search heuristic for constraint relationships
- Investigate conditional statements

Practical

- Apply constraint relationships to an optimisation problem in power management systems
- Learning/abductive reasoning for preference elicitation