Constraint Relationships for Soft Constraints

Alexander Schiendorfer, Jan-Philipp Steghöfer, Alexander Knapp, Florian Nafz and Wolfgang Reif

Abstract We introduce *constraint relationships* as a means to define qualitative preferences on the constraints of soft constraint problems. The approach is aimed at constraint satisfaction problems (CSPs) with a high number of constraints that make exact preference quantizations hard to maintain manually or hard to anticipate — especially if constraints or preferences change at runtime or are extracted from natural language text. Modelers express preferences over the satisfaction of constraints with a clear semantics regarding preferred tuples without assigning priorities to concrete domain values. We show how a CSP including a set of constraint relationships can linearly be transformed into a k-weighted CSP as a representative of c-semirings that is solved by widely available constraint solvers and compare it with existing techniques. We demonstrate the approach by using a typical example of a dynamic and interactive scheduling problem in AI.

1 Introduction

Numerous real world problems in AI including planning, scheduling, or resource allocation have been addressed successfully using the generic framework of constraint satisfaction. One has to identify the decision variables of a problem as well as constraints that regulate legal assignments.

While classical constraints can be considered "law-like" formulations that *must not* be violated, preferences constitute desired properties a solution *should* have. For instance, in a university timetabling problem, a constraint states that a professor can never give two lectures at the same time. In addition, we might *prefer* solutions that do not include Friday afternoon lectures.

In a perfect world, we are able to find solutions that satisfy all constraints and preferences. However, real-world problems tend to become too rigid as problems become over-constrained due to additional constraints that represent preferences. Pioneering approaches to this problem either change the problem by relaxing existing constraints by adding domain values as in Partial CSP [12] or look for solutions that fulfill as many

Alexander Schiendorfer
Institute for Software & Systems Engineering, Augsburg University, Germany
e-mail: `alexander.schiendorfer@student.uni-augsburg.de`

Jan-Philipp Steghöfer, Alexander Knapp, Florian Nafz, and Wolfgang Reif
Institute for Software & Systems Engineering, Augsburg University, Germany
e-mail: `{steghoefer,knapp,nafz,reif}@informatik.uni-augsburg.de`

constraints as possible as in MaxCSP [14]. Usually, we are interested in assignments that satisfy all mandatory constraints and moreover satisfy preferences as well as possible. We present a qualitative formalism that enables to make statements such as "We prefer a solution that satisfies constraint X to another one that fulfills both Y and Z but violates X".

Our contribution consists of two parts. First, we propose constraint relationships that provide a useful and time-saving modeling and elicitation tool to abstractly denote preferences. We illustrate their usage by analyzing scenarios for a typical example of the scheduling problem. Second, we give a transformation into a k-weighted CSP that respects the dominance properties we formalized and can be used with off-the-shelf constraint solvers therefore bridging between the preference elicitation domain and the world of soft constraints.

1.1 Constraint Relationships

Constraint relationships constitute a qualitative "is more important than"-relation over constraints to denote which constraints should rather be satisfied and which ones can be dropped if necessary. For instance, stating "having a nice view" is more important than "spending less than three hours in a bus" models decisions where — other parameters being equal — plans satisfying the duration constraint but not the landscape requirement are considered worse than those offering interesting landscapes. We argue that this is a useful and realistic approach that enables an easy-to-use formalism for special application areas and positions itself among the AI formalisms presented in Sect. 1.2. Constraint relationships are targeted at problems with many constraints where preference levels are hard to maintain manually, or dynamic constraint satisfaction problems with frequently changing constraints and preferences. Typical applications facing these circumstances are multi-agent systems with incomplete preferences [19].

In some applications (e.g., [10] constructed a utility model of goals of their agents that is transferred into a numerical utility function), it is considered more natural to denote preferences in a qualitative rather than a quantitative way which results in an increasing interest in qualitative formalisms in AI [7]. For instance, vague preference statements formulated in natural language can be easier formulated in a qualitative way.

On the quantitative side, existing soft constraint approaches work well if the CSP is known completely at design time. Designers can assign weights, categorize constraints into a hierarchy or assign coherent coefficients that yield acceptable solutions. However, assigning penalties that express preferences correctly is not straightforward in systems of several dozen constraints, in open-world scenarios, or dynamic constraint satisfaction problems [18]. Constraints may be added or removed at runtime to generate new CSPs [9, 2]. New constraints can change the system considerably and assigning penalties can become difficult, usually requiring manual intervention.

The decision whether a constraint is hard or soft and how important it is results from a "preference elicitation" process (see, e.g., [13]) and uses techniques from decision analysis. Upon identifying an over-constrained problem, a human expert has to provide information to distinguish constraints and to establish preferences on them. Such an approach makes large problems difficult to handle. It has been pointed out that such a fixed quantitative model performs poorly in situations where autonomous agents face new situations and need to change their preferences [10]. Numerical mappings lose information about underlying rationales hindering their reuse.

Therefore, we suggest the definition of *constraint relationships* as the core of the elicited preference model being abstracted from numerical values. This allows for recombination, changes, additions of new preferences as well as compositions of multiple constraint relationship sets at runtime in, e.g., a hierarchy of related CSPs.

To use our preference semantics with existing solvers, we give a transformation that preserves the desired dominance property (see Sect. 2.1) into a k-weighted CSPs which is a representative of c-semirings discussed in Sect. 1.2. The resulting model can be used with standard weighted CSP solvers or solved with a constraint satisfaction optimization approach. The entire process is tool supported and can be fully automated, enabling its use in dynamic environments.

1.2 Preference Formalisms in AI

In the past decades, several AI preference formalisms have been devised to cope with this problem [17]. Among these are several flavors of soft constraints such as Fuzzy CSP [11, 13] or Weighted CSP (WCSP) [22] as well as other mechanisms like conditional preference networks (CP-nets) [5], constraint hierarchies [4] or Bayesian nets.

Generic frameworks for soft constraints such as c-semirings [3] or valued constraints [21] have been proposed to design generic algorithms and prove useful properties over a common structure. Assignments are labeled with preference levels or violation degrees that are combined using a specific operator to find preferred solutions. Formally, a *c-semiring* is given by a quintuple $\langle E, +_s, \times_s, \mathbf{0}, \mathbf{1} \rangle$ where E is the set of preference levels; $+_s$ is a binary operation closed in E used to compare assignments (defining $e \geq_s e' \leftrightarrow e +_s e' = e$) for which $\mathbf{0}$ is a neutral element (i.e., $e +_s \mathbf{0} = e$) and $\mathbf{1}$ is an annihilator (i.e., $e +_s \mathbf{1} = \mathbf{1}$); and $\times_s$ is a binary operation also closed in E used to combine preference levels where $\mathbf{0}$ is an annihilator and $\mathbf{1}$ a neutral element.

In WCSPs, a cost function is provided for each constraint. The function assigns a weight from $\mathbb{R}_{\geq 0}$ to each tuple of values. Solving a WCSP then consists of minimizing the sum of weights of all violated constraints. This basic form has been refined to *k-weighted constraint networks* that include a special weight k for hard constraints, meaning that assignments over the threshold k are considered unacceptable [15]. The resulting c-semiring is then defined as $\langle 0..k, \min, +^k, k, 0 \rangle$ (with $0..k = \{0, \ldots, k\}$ and $x +^k y = \min\{k, x + y\}$) which is the instance we will use to denote the preference semantics of constraint relationships.

In *Fuzzy CSPs* [11, 13], it is possible to express the satisfaction degree of an assignment with respect to a constraint according to the theory of fuzzy sets. The individual membership functions of the constraints (representing how strongly an assignment is an element of the constraint's fuzzy set of allowed assignments) are usually conjoined by taking the minimum of their satisfaction degrees. Thus, an assignment is judged based on the worst satisfaction degree although other operators have been proposed [17].

Outside the class of soft constraint problems, in *constraint hierarchies* [4], constraints are categorized into strict levels, represented as sets $H_0, \ldots, H_n$ in which constraints in H_i take precedence over constraints in H_{i+1} (that is, level i dominates level $i+1$). Valid solutions fulfill all constraints in H_0 and as many as possible in the dominated levels. Constraint hierarchies offer a similar design paradigm to constraint relationships. Although constructs like partial order hierarchies and different comparators have been proposed, this formalism is suited well for problems that incorporate a "totalitarian" semantics [7] — i.e., it is definitely better to satisfy a more important constraint rather than numerous less important ones. We argue that different semantics are required for other, more "egalitarian" problem classes. In constraint relationships, it is possible to leave out definite importance decisions meaning that the modeler does not prefer one constraint over the other — hierarchies only enable this for constraints on the same level. Furthermore, we show in Sect. 2.2 that constraint relationships can express a particular class of constraint hierarchies, viz. *locally-predicate-better* hierarchies, but model additional solution preferences.

CP-nets [6] are a qualitative tool to represent preferences over assignments. They intend to reduce the complexity of all possible preference orderings by structuring the variables according to their mutual impact.

Concretely, it is assumed, that some variable assignments do not affect the preference order on the domain values of others — whereas others do. Given this structural information about influences, a decision maker is asked to explicitly specify her preferences over the values of a variable X for *each* assignment to its *parent* variables – i.e. the ones that affect the preference on the domain of X. Intuitively, whichever solution is preferred for a parent variable thus influences the best solution for X. Hence, a set of total orders on the values of finite domains is stored for every variable in a conditional preference table that results in the most general case in a preorder on the solutions [20].

This approach allows capturing conditional preferences over explicit domain values such as “If lunch is vegetarian, prefer white wine to red wine” and “If lunch is meat, prefer red wine to white wine”. It is possible to transform a CP-net into a classical constraint problem that yields the same set of optimal solutions. However this is not true for the solution ordering in general [17]. While many real-world examples can be adequately modeled using CP-nets if user preferences on the actual values can be elicited [6], we argue that for complex constraint networks with infinite domains it is easier to make generalizing statements [7] that refer to a coarser level of granularity — preferences on constraints rather than on domain values for single variables. As the number of constraints is arguably lower than the number of variables and possible domain values for larger problems, the definition of preferences over constraints simplifies the preference elicitation problem considerably. Whereas CP-nets offer efficient solving algorithms due to the extensional structure of variables and values, our tool is intended to work with intensional constraints as well. CP-nets work by defining total orders over domain values for a *single* variable, constraint relationships can induce an order over solutions by prioritizing the constraints of a CSP. A comparison to CP-nets can be found in Sect. 2.3.

Preliminaries

We follow and extend the definitions for classical constraint networks and soft constraints used in [17].

A *constraint network* $\langle X, D, C\rangle$ is given by

- a finite set $X = \{x_1, \dots, x_n\}$ of *n variables*;
- a set $D = \{D_1, \dots, D_n\}$ of corresponding *domains* such that x_i takes values from D_i; and
- a finite set C of *constraints*. Each $c \in C$ is given by a pair (r, v) where $v \subseteq X = sc(c)$ represents the scope of the constraint and $r \subseteq \prod_{x_i \in v} D_i = rl(c)$ is a relation containing the allowed combinations for c.

We distinguish between *hard constraints* $C_h \subseteq C$ and *soft constraints* $C_s \subseteq C$, such that $C_h \cup C_s = C$ and $C_h \cap C_s = \emptyset$.

Assignments are tuples $t_V \in \prod_{x_i \in V} D_i$ with $V \subseteq X$. If $W \subseteq V$, $t_V[W]$ returns the *projection* of a tuple containing only elements in W. An assignment is *complete* if $V = X$, and *partial* otherwise. A constraint $c \in C$ is *fully assigned* by t_V if $sc(c) \subseteq V$. An assignment is *consistent with* c if c is fully assigned by t_V and $t_V[sc(c)] \in rl(c)$; we then write $t_V \models c$. An assignment is a *solution* if $\forall c \in C_h : t_V \models c$. We often omit the variables from assignments and write t instead of t_V.

A *weighted constraint network* $\langle X, D, C, w\rangle$ is given by a constraint network $\langle X, D, C\rangle$ and a *weighting function* and $w : C \to \mathbb{R}_{\geq 0}$. The weight of an assignment t_V is the combined weight of all unsatisfied constraints: $w(t_V) = \sum_{c \in C : t_V \not\models c} w(c)$. The purpose of a weighted CSP is to find assignments having minimal weight.

We write $C'_1 \uplus C'_2$ to denote the *disjoint union* of C'_1 and C'_2, where we require that $C'_1 \cap C'_2 = \emptyset$.

A binary relation $Q \subseteq M \times M$ on a set M is *asymmetric* if $(m, m') \in Q$ implies that $(m', m) \notin Q$; it is *transitive* if $(m, m') \in Q$ and $(m', m'') \in Q$ implies $(m, m'') \in Q$; it is a *partial order relation* if it is asymmetric and transitive. The *transitive closure* of Q, denoted by Q^+, is inductively defined by the rules that (a) if $(m, m') \in Q$, then $(m, m') \in Q^+$ and (b) if $(m, m') \in Q$ and $(m', m'') \in Q^+$, then $(m, m'') \in Q^+$.

2 Constraint Relationships

A set of *constraint relationships* for the soft constraints C_s of a constraint network $\langle X, D, C\rangle$ is given by a binary asymmetric relation $R \subseteq C_s \times C_s$ whose transitive closure R^+ is a partial order relation. We write $c' \prec_R c$ or $c \succ_R c'$ iff $(c, c') \in R$ to define c to be *more important* than c', analogously for R^+. If $c' \prec_R c$ we call c' a *direct predecessor*, if $c' \prec_{R^+} c$ a *transitive predecessor* of c. Moreover, we refer to the *constraint relationship graph* as the directed graph spanned by $\langle C_s, R\rangle$.

2.1 Semantics of Dominance Properties

We have defined constraint relationships syntactically as a relation that denotes some constraint being "more important" than another one. However, we need to express *how* much more important a constraint is than another one in order to address questions such as "Is it better to satisfy a more important constraint than *all* its less important predecessors?". Concretely, we examine ways to lift the binary relation over soft constraints to sets of soft constraints that are violated by an assignment. Such a *violation set* is denoted by capitalizing the letter used for the assignment; i.e., for some assignment t its violation set is $T = \{c \in C_s \mid t \not\models c\}$.

We consider several possibilities for worsening a violation set and will use $T \longrightarrow_R^p U$ to express that T is worsened to U by using the strategy or *dominance property p*. All strategies share two generic rules that we present first. On the one hand, a set of violated constraints T gets worse if some additional constraint c is violated:

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

On the other hand, worsening two independent parts of a violation set leads to a worsening of the whole violation set: If T_1 is worsened to U_1 and T_2 is worsened to U_2, then $T_1 \uplus T_2$ is also worsened to $U_1 \uplus U_2$:

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

In fact, we can derive

$$T \longrightarrow_R^p T \uplus \{c_1, \dots, c_k\} \tag{W1'}$$

We have $T \longrightarrow_R^p T \uplus \{c_1\}$ by (W1) and $\emptyset \longrightarrow_R^p \{c_2\}$ again by (W1), and thus $T \longrightarrow_R^p T \uplus \{c_1, c_2\}$ by (W2), from which the result follows by induction.

We now introduce three particular dominance properties. In the first approach, violating a less important constraint rather than an important one should be considered *better — ceteris paribus*. We call this criterion *single predecessor dominance* (SPD):

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

For instance, if $C_s = \{\mathsf{a}, \mathsf{b}\}$ and $\mathsf{a} \succ_R \mathsf{b}$ (constraint a is considered more valuable than constraint b), $\{\mathsf{a}\} \longrightarrow_R^{\text{SPD}} \{\mathsf{a}, \mathsf{b}\}$ by (W1) (with $p = $ SPD), $\{\mathsf{b}\} \longrightarrow_R^{\text{SPD}} \{\mathsf{a}\}$ by (SPD). Since SPD does not have a single constraint dominate a set of others it is well suited for "egalitarian" problems. It is therefore similar to a MaxCSP instance where we are interested in satisfying a large number of constraints rather than discriminating strongly by their individual importance.

However, a stronger notion is needed when some constraints contribute more to the quality of a solution than a whole set of others — in particular the constraints that are explicitly denoted less important. This property is called *direct predecessors dominance* (DPD) and is motivated by the fact that human preference decisions can be intransitive [1]:

$$T \uplus \{c_1, \dots, c_k\} \longrightarrow_R^{\mathrm{DPD}} T \uplus \{c'\} \quad \text{if } \forall c \in \{c_1, \dots, c_k\} : c \prec_R c' \tag{DPD}$$

For a minimal example, consider $C_s = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$ and $\mathsf{a} \succ_R \mathsf{b}$, $\mathsf{a} \succ_R \mathsf{c}$. Then violating a is more detrimental to a solution than any combination of b and c, e.g., $\{\mathsf{b}, \mathsf{c}\} \longrightarrow_R^{\mathrm{DPD}} \{\mathsf{a}\}$. It follows by definition that $T \longrightarrow_R^{\mathrm{SPD}} U$ implies $T \longrightarrow_R^{\mathrm{DPD}} U$.

The most "hierarchical" notion, *transitive predecessors dominance*, consists of extending DPD to include transitive predecessors as well. It is motivated by the natural extension of constraint relationships R to its transitive closure R^+ to obtain a partial order and the ability to express a subset of constraint hierarchies with constraint relationships. Note that this property could also be achieved by using DPD rules and R^+ explicitly in the model:

$$T \uplus \{c_1, \dots, c_k\} \longrightarrow_R^{\mathrm{TPD}} T \uplus \{c'\} \quad \text{if } \forall c \in \{c_1, \dots, c_k\} : c \prec_{R^+} c' \tag{TPD}$$

If $C_s = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$ and $\mathsf{a} \succ_R \mathsf{b}$, $\mathsf{b} \succ_R \mathsf{c}$, then $\{\mathsf{b}, \mathsf{c}\} \longrightarrow_R^{\mathrm{TPD}} \{\mathsf{a}\}$, but also $\{\mathsf{c}\} \longrightarrow_R^{\mathrm{TPD}} \{\mathsf{b}\}$. Again, by definition $T \longrightarrow_R^{\mathrm{DPD}} U$ implies $T \longrightarrow_R^{\mathrm{TPD}} U$. As mentioned before, $T \longrightarrow_R^{\mathrm{TPD}} U$ iff $T \longrightarrow_{R^+}^{\mathrm{DPD}} U$.

Each relation $\longrightarrow_R^p$ over assignments induced by the used semantics describes *how* an assignment is worsened. It only is a partial order if the generating relation (R or R^+) already is a partial order. In general, the relation does not need to be transitive as this property is not required for R, which corresponds to the fact that preference relations need not generally be transitive [1]. Consider for example $C_s = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$, $\mathsf{a} \succ_R \mathsf{b}$, $\mathsf{b} \succ_R \mathsf{c}$, and an SPD semantics; then $\{\mathsf{c}\} \longrightarrow_R^{\mathrm{SPD}} \{\mathsf{b}\}$ and $\{\mathsf{b}\} \longrightarrow_R^{\mathrm{SPD}} \{\mathsf{a}\}$, but not $\{\mathsf{c}\} \longrightarrow_R^{\mathrm{SPD}} \{\mathsf{a}\}$ since $\mathsf{a} \not\succ_R \mathsf{c}$.

We can enforce partial orders on assignments for each dominance property $p \in \{\mathrm{SPD}, \mathrm{DPD}, \mathrm{TPD}\}$, denoted by $t >_R^p u$ and to be read as "t is better than u", using $T\ (\longrightarrow_R^p)^+\ U$ (meaning repeated sequential application of the rules); we will prove the asymmetry of these transitive closures in Sect. 3 using weights.

2.2 Connection to Constraint Hierarchies

Constraint hierarchies [4] offer different *comparators* to discriminate solutions based on their satisfaction degree of constraints at different levels given by an error function and additional weights for constraints. Comparators are divided into *locally better* and *globally better*. Locally better compares based on the error functions only, whereas globally better predicates also take constraint-specific weights into account. Both comparator types are lexicographic in the sense that an ordering is found at the first level of the hierarchy where one solution performs better than the other.

Error functions are either *predicate* functions, i.e., $e(c,t) = 0$ if $t \models c$, and 1 otherwise; or *metric* functions that give a continuous degree of violation (e.g., for $c \triangleq (X = Y)$, $e(c,t)$ could return the difference between the valuations of the variables X and Y in t). Since as of now our approach is not concerned with metric error functions or combinations with existing weights, we restrict ourselves to comparison with *locally predicate better* (LPB) and show that these hierarchies can be encoded in constraint relationships. We then show that constraint relationships generalize LPB-hierarchies by providing an example that cannot be expressed in LPB-hierarchies.

First, consider the definition of LPB given by a constraint hierarchy $H = \{H_0, \dots, H_n\}$. The operator $>_{LPB}$ compares two solutions t and u (the constraints in H_0 are taken to be the hard constraints) and $t >_{LPB} u$ should be read as "t is better than u"; it is defined by

$$\begin{aligned} t >_{LPB} u \leftrightarrow \exists k > 0 : (&\forall i \in 1..k-1 : \forall c \in H_i : e(c,t) = e(c,u)) \wedge \\ &(\forall c \in H_k : e(c,t) \leq e(c,u)) \wedge (\exists c \in H_k : e(c,t) < e(c,u)) \end{aligned}$$

Our encoding of the constraint hierarchy H in constraint relationships R_H is defined as follows: $C_h = H_0$, $C_s = \bigcup_{i \in 1..n} H_i$, and

$$c \succ_{R_H} c' \leftrightarrow c \in H_i \wedge c' \in H_{i+1} \ .$$

We write T_i for all constraints in hierarchy level i that are violated by an assignment t, i.e., $T_i = \{c \in H_i \mid t \not\models c\}$ and, analogously, U_i for an assignment u; we abbreviate $\bigcup_{k \leq i \leq l} T_i$ by $T_{k..l}$.

Theorem 1. *If $t >_{LPB} u$, then $T \longrightarrow^{\mathrm{TPD}}_{R_H} U$.*

Proof. Observe that $e(c,t) < e(c,u)$ iff $t \models c \wedge u \not\models c$; and $e(c,t) \leq e(c,u)$ iff $u \models c \rightarrow t \models c$.

Let $t >_{LPB} u$; we have to show that T is worsened to U by application of TPD-rules. Let $k > 0$ be such that (*) $\forall i \in 1..k-1 : \forall c_i \in H_i : t \models c_i \leftrightarrow u \models c_i$, (**) $\forall c_k \in H_k : u \models c_k \rightarrow t \models c_k$, and let $c \in H_k$ such that $t \models c \wedge u \not\models c$. By (*) we have that $T_{1..k-1} = U_{1..k-1}$. Furthermore, $T_{1..k} \subseteq U_{1..k} \setminus \{c\}$ since $T_{1..k-1} = U_{1..k-1}$, $\forall c_k \in H_k : t \not\models c_k \rightarrow u \not\models c_k$ by (**), and $c \notin T_{1..k}$. In particular, $T_{1..k} \subseteq U_{1..k} \setminus \{c\} \subseteq U_{1..n} \setminus \{c\}$. If $T_{1..k} = U_{1..n} \setminus \{c\}$, then $T = T_{1..k} \uplus T_{k+1..n} \longrightarrow^{\mathrm{TPD}}_{R_H} T_{1..k} \uplus \{c\} = U$ by (TPD), since all constraints in $T_{k+1..n}$ are transitively dominated by $c \in H_k$. If $T_{1..k} \subsetneq U_{1..n} \setminus \{c\}$, then $T_{1..k} \longrightarrow^{\mathrm{TPD}}_{R_H} U_{1..n} \setminus \{c\}$ by (W1'); applying rule (TPD) in (W2), we again have $T = T_{1..k} \uplus T_{k+1..n} \longrightarrow^{\mathrm{TPD}}_{R_H} (U_{1..n} \setminus \{c\}) \uplus \{c\} = U$. $\square$

Conversely, Fig. 1a shows a constraint relationship problem that is not expressible in LPB hierarchies. Let $H : \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{e}\} \rightarrow \mathbb{N} \setminus \{0\}$ be a mapping from the constraints to their respective hierarchy levels. We consider solutions that only satisfy *one* constraint and violate all others and write a for "a solution satisfying only a". We show that every admissible choice of H introduces too much ordering: The constraint relationships require a to be better than b which in turn should be better than c, thus we have to have $H(\mathsf{a}) < H(\mathsf{b}) < H(\mathsf{c})$. Since we expect a to be more important than d as well, but require b and d to be incomparable, $H(\mathsf{d})$ has to be equal to $H(\mathsf{b})$. Similarly, $H(\mathsf{e})$ has to be $H(\mathsf{c})$ as e and c should be incomparable. But then b would be better than e, a relation that is explicitly not modeled in the underlying constraint relationships.

Future work includes metric error functions that could be added lexicographically to the existing dominance properties. Combination with existing weights (as would be required for globally-better comparators) is not intended as the approach targets problems that are modeled qualitatively.

2.3 Connection to CP-Nets

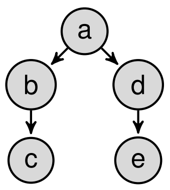
CP-nets [6] specify total orders over the domain of a variable depending on an assignment to other variables in a so-called conditional preference table. Concretely, a preference statement for a variable y is written as:

$$x_1 = d_1, \dots, x_n = d_n : y = w_1 \succ \cdots \succ y = w_k$$

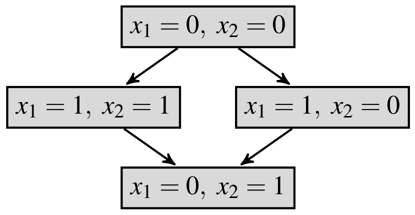
where $x_1,\ldots,x_n$ are the *parent* variables of y and $w_1,\ldots,w_k$ are all domain values of y given in a total order $\succ$. Such an order needs to be specified for all assignments to $x_1,\ldots,x_n$. A preference statement should be interpreted as "Given that $x_1 = d_1,\ldots,x_n = d_n$, all other variables being equally assigned, prefer a solution that assigns w_i to y over one that assigns w_j to y iff $i > j$" which is the *ceteris paribus* assumption. The change of value for y from w_i to w_j is then called a "worsening flip". A complete assignment t to the variables of a CP-net is preferred to another one, say t' , if t' can be obtained from t via a sequence of worsening flips [17].

On the one hand, the induced "better-as" relation on assignments need not be a partial order since cycles may arise [5]. By contrast, constraint relationships always lead to a partial order $>^p_R$ on assignments. On the other hand, CP-nets cannot express all partial orders on assignments [20]. Consider the minimal example depicted in Fig. 1b: $X = \{x_1,x_2\}$, $D_1 = D_2 = \{0,1\}$. The proposed solution order cannot be expressed in CP-nets since $x_1 = 1,\ x_2 = 0$ and $x_1 = 1,\ x_2 = 1$ differ only by the assignment of x_2 and have to be comparable because of the total order requirement and *ceteris paribus* semantics in CP-nets. But this solution ordering is easily expressible in constraint relationships defining a constraint for each possible assignment.

Thus, the two frameworks are incomparable regarding the solution order. An extension, however, of the constraint relationship approach with conditional statements as in CP-nets might turn out to be fruitful.



(a) A set of constraint relationships not expressible in LPB-hierarchies



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

Fig. 1: Constraint Relationships compared to other formalisms

3 Transforming Constraint Relationships into Weighted CSPs

Once the constraint problem and its constraint relationships are defined and the dominance property is chosen, we want to apply this information by transforming it into a k-weighted CSP that we can solve with standard solvers or special purpose weighted CSP-algorithms.

For a given CSP $\langle X,D,C\rangle$, a set of constraint relationships R, and a dominance property $p \in \{\mathrm{SPD},\mathrm{DPD},\mathrm{TPD}\}$, we devise a weighting function $w^p_R : C \to \mathbb{N} \setminus \{0\}$ which we prove to be strictly monotonic w.r.t. to the dominance property p for soft constraints in the sense that $\sum_{c\in T} w^p_R(c) < \sum_{c\in U} w^p_R(c)$ if $t >^p_R u$. For each hard constraint $c \in C_h$, we set $w^p_R(c) = k^p_R$ with $k^p_R = 1 + \sum_{c\in C_s} w^p_R(c)$; therefore a solution of $\langle X,D,C\rangle$ satisfying only hard constraints still has a combined weight less than k^p_R. In particular, we thus can use the c-semiring $\langle 0..k^p_R, \min, +^{k_R^p}, k^p_R, 0\rangle$ for solving the soft-constraint problem, where solvers are readily available. However, it has to be noted that using such weights all solutions become comparable due to the totality of the order on the accumulated weights, though different solutions may show the same accumulated weight. This "defect" is not an inherent property of constraint relationships and their induced orderings on solutions, but is an artifact of the transformation into k-weighted CSPs. It would be interesting to find a c-semiring for representing constraint relationships that does not introduce such artificial orderings.

3.1 Concrete Weight Functions

Each w_R^p will be defined recursively relying on the weights of predecessors w.r.t. R to have been already calculated. This is well defined since C and hence R are finite and R^+ is a partial order (i.e., the graph of R^+ is acyclic). Algorithmically, the weights can be determined in a bottom-up fashion or using a depth-first strategy.

For establishing the respective strict monotonicity properties, let $W_R^p(T) = \sum_{c \in T} w_R^p(c)$; we then have to prove $W_R^p(T) < W_R^p(U)$ if $t >_R^p u$. Since $>_R^p$ is defined via $(\longrightarrow_R^p)^+$ and all dominance properties are defined inductively by rules, it suffices to demonstrate strict monotonicity for each rule application: $W_R^p(T) < W_R^p(U)$ if $T \longrightarrow_R^p U$.

Rule (W1) says that $T \longrightarrow_R^p T \uplus \{c\}$; and indeed, $W_R^p(T) < W_R^p(T \uplus \{c\})$, since all weights are in $\mathbb{N} \setminus \{0\}$. Rule (W2) has the premises $T_i \longrightarrow_R^p U_i$ for $i \in \{1,2\}$; these amount to the assumptions $W_R^p(T_i) < W_R^p(U_i)$, from which we can conclude that $W_R^p(T_1 \uplus T_2) = W_R^p(T_1) + W_R^p(T_2) < W_R^p(U_1) + W_R^p(U_2) = W_R^p(U_1 \uplus U_2)$. It thus remains to prove the strict monotonicity of (SPD), (DPD), and (TPD).

Single predecessor dominance. For SPD, we propose the function that takes the maximum weight of its predecessors and adds 1 (we take $\max(\emptyset)$ to be 0):

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

This mapping is indeed strictly monotonic for applications of rule (SPD): Let $T \uplus \{c\} \longrightarrow_R^{\mathrm{SPD}} T \uplus \{c'\}$ with $c \prec_R c'$. Then $w_R^{\mathrm{SPD}}(c) < w_R^{\mathrm{SPD}}(c')$ and hence $W_R^{\mathrm{SPD}}(T \uplus \{c\}) = W_R^{\mathrm{SPD}}(T) + w_R^{\mathrm{SPD}}(c) < W_R^{\mathrm{SPD}}(T) + w_R^{\mathrm{SPD}}(c') = W_R^{\mathrm{SPD}}(T \uplus \{c'\})$.

Direct predecessors dominance. For DPD we take the sum of weights of all predecessors and add 1 (summation over an empty index set is taken to be 0):

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

Rule (DPD) requires that violating a single constraint is worse than violating all its direct predecessors and hence this weight assignment assures that the weight of a constraint is strictly greater than the sum of the set of *all* its direct predecessors. In fact, for strict monotonicity, let $T \uplus \{c_1, \ldots, c_k\} \longrightarrow_R^{\mathrm{DPD}} T \uplus \{c'\}$ with $c_i \prec_R c'$ for all $1 \leq i \leq k$. Then $W_R^{\mathrm{DPD}}(\{c_1, \ldots, c_k\}) = \sum_{1 \leq i \leq k} w_R^{\mathrm{DPD}}(c_i) < w_R^{\mathrm{DPD}}(c) = W_R^{\mathrm{DPD}}(\{c'\})$, by definition of w_R^{DPD}, and $W_R^{\mathrm{DPD}}(T \uplus \{c_1, \ldots, c_k\}) < W_R^{\mathrm{DPD}}(T \uplus \{c'\})$.

Transitive predecessors dominance. Analogous to the DPD case, a TPD preserving weight assignment function w_R^{TPD} needs to make sure that $w_R^{\mathrm{TPD}}(c) > \sum_{c' \in C_s : c \succ_{R^+} c'} w_R^{\mathrm{TPD}}(c')$. Since, as mentioned in Sect. 2.1, TPD is DPD for R^+, the function $w_{R^+}^{\mathrm{DPD}}$ would suffice. However, we can avoid computing the transitive closure of R by using the following function that also only depends on the direct predecessors:

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

We can establish that $w_R^{\mathrm{TPD}}(c) \geq 1 + \sum_{c' \in C_s : c \succ_{R^+} c'} w_R^{\mathrm{TPD}}(c')$ for all $c \in C_s$ with this definition by induction over the number of transitive predecessors of c: If $\{c' \in C_s \mid c \succ_{R^+} c\} = \emptyset$, then $w_R^{\mathrm{TPD}}(c) = 1$ and thus the claim holds. Let $\{c' \in C_s \mid c \succ_{R^+} c\} \neq \emptyset$ and assume that $w_R^{\mathrm{TPD}}(c') \geq 1 + \sum_{c'' \in C_s : c' \succ_{R^+} c''} w_R^{\mathrm{TPD}}(c'')$ holds for all $c' \in C_s$ such that $c \succ_R c'$. Then

$$\begin{aligned} w_R^{\mathrm{TPD}}(c) &= 1 + \textstyle\sum_{c' \in C_s : c \succ_R c'} (2 \cdot w_R^{\mathrm{TPD}}(c') - 1) \\ &= 1 + \textstyle\sum_{c' \in C_s : c \succ_R c'} (w_R^{\mathrm{TPD}}(c') + (w_R^{\mathrm{TPD}}(c') - 1)) \\ &\geq 1 + \textstyle\sum_{c' \in C_s : c \succ_R c'} (w_R^{\mathrm{TPD}}(c') + \sum_{c'' \in C_s : c' \succ_{R^+} c''} w_R^{\mathrm{TPD}}(c'')) \\ &\geq 1 + \textstyle\sum_{c' \in C_s : c \succ_{R^+} c'} w_R^{\mathrm{TPD}}(c') \end{aligned}$$

where the last inequation holds since every summand of the last line is a summand of the next-to-last line.

In summary, we have

Theorem 2. *If $t >_R^p u$, then $W_R^p(T) < W_R^p(U)$ for $p \in \{\mathrm{SPD}, \mathrm{DPD}, \mathrm{TPD}\}$.* □

In particular, $>_R^p$ is asymmetric for $p \in \{\mathrm{SPD}, \mathrm{DPD}, \mathrm{TPD}\}$ since the order $<$ on the weights is asymmetric: If we would have $t >_R^p u$ and $u >_R^p t$, then $W_R^p(T) < W_R^p(U)$ and $W_R^p(U) < W_R^p(T)$ by the theorem, which is impossible.

3.2 Solving the Weighted CSP

Once the weights of the individual constraints have been established, the problem presents as a normal Weighted CSP and can be solved with appropriate algorithms [21, 16]. Alternatively, the original constraint satisfaction problem can be restated as a constraint satisfaction optimization problem. For this purpose, each constraint is reformulated as follows:

$$\hat{c}_i : c_i \vee p_i = w_R^p(c_i)$$

This allows a constraint optimizer to either fulfill c_i or set $0 \neq p_i \in \mathbb{N}$, the penalty for violating c_i, to the weight of the corresponding constraint. The objective is then to minimize the sum of penalties over all constraints in C with respect to the threshold k_R^p:

$$\text{minimize } \textstyle\sum_{c_i \in C} p_i \text{ such that } \sum_{c_i \in C} p_i < k_R^p$$

A constraint relationship problem formulated this way can be solved with commercial or open source software, such as the IBM ILOG CPLEX Optimizer [8] which we used for a prototypical implementation. However, the approach described in this paper is independent from the concrete language the constraints or problems are formulated in or the solver that is used.

4 Example Scenario – Ski Day Planner

We demonstrate a typical application of constraint relationships using a simplified fictional yet plausible real-world scenario. First, we show how varying sets of relationships lead to different preferred assignments. Then we discuss how changing constraints or preferences can be included.

4.1 Constraints & Preferences

The following example illustrates the effects of different constraint relationship graphs in an interactive decision-support problem. Concretely, consider an application that provides guidance to travelers exploring a new ski area by offering a plan for a ski day. The problem might include real-time lift occupancy data as well as weather forecasts and snow conditions. Each skier has different priorities that can be set interactively.

Assume the following soft constraints are defined on the set of possible tours (that need to respect hard constraints such as weather induced blockages, timing issues, restaurant opening hours and daylight time):

- Avoid black slopes (ABS): Beginners might want to avoid difficult (marked "black") slopes.
- Variety (VT): Different slopes need to be explored rather than staying on the same tracks.
- Fun-park (FP): Freestyle fans want to include a fun-park.
- Little Wait (LW): Impatient visitors prefer not to wait too long at a lift.
- Only Easy Slopes (OE): People can restrict their tours to easy ("blue") slopes only.
- Lunch Included (LI): Whereas some travelers enjoy the comfort of a good mountain dish, others prefer to spend their day out without longer rest.

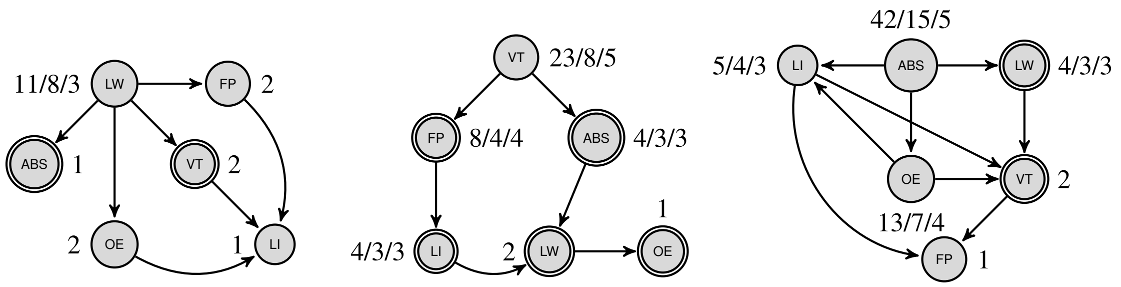
For clarity, we abstract from technical details such as concrete tuple representations and leave hard constraints aside by assuming the following three assignments satisfy all hard constraints but differ in their performance on soft constraints.

- $t_X^{(1)} \models \{\mathsf{LW}, \mathsf{OE}, \mathsf{LI}, \mathsf{FP}\} \wedge t_X^{(1)} \not\models \{\mathsf{VT}, \mathsf{ABS}\}$
- $t_X^{(2)} \models \{\mathsf{VT}\} \wedge t_X^{(2)} \not\models \{\mathsf{FP}, \mathsf{ABS}, \mathsf{LW}, \mathsf{LI}, \mathsf{OE}\}$
- $t_X^{(3)} \models \{\mathsf{OE}, \mathsf{FP}, \mathsf{ABS}, \mathsf{LI}\} \wedge t_X^{(3)} \not\models \{\mathsf{VT}, \mathsf{LW}\}$

Assume three personas as prototypical customers.

- Skier *A* is rather impatient, skilled in skiing, wants to explore a fun-park but is not particularly afraid of difficult slopes or needs lunch since he perceives skiing primarily as workout.
- Boarder *B* is an explorer, she prefers to see a large number of different slopes (not necessarily black ones) but accepts to wait longer if that is required.
- Rookie *C* just started skiing and therefore wants to avoid black slopes and appreciates a tour consisting of easy slopes. He prefers a good meal to extraordinary adventures.

Figure 2 depicts corresponding constraint relationship graphs extracted from this description and Tab. 1 shows how the sample assignments are evaluated using the exemplary relationship graphs. Every relationship graph yields a different winner and we need to judge whether those decisions fit the preferences. *A* indicated little interest in variety, avoiding black slopes or easy tracks and got the only assignment that does not require waiting. Similarly, we get a match for *B*'s requirements and do not force *C* to take difficult slopes. The calculated assignment winners thus make sense and show how the different graphs influence the decision process to a strong degree. Interestingly for *B*, the selected dominance property affects the preferred solution. Since $t_X^{(2)}$ only satisfies VT and violates the 5 other constraints, it is considered worse than $t_X^{(3)}$ in SPD and DPD semantics. However, as VT is the most important constraint for *B* and is only satisfied by $t_X^{(2)}$, in a TPD semantics this solution is still preferred over the others satisfying more constraints.



(a) Graph of skier *A* (b) Graph for boarder *B* (c) Graph for rookie *C*

Fig. 2: The constraint relationship graphs for each persona. Double borders indicate that this constraint was violated in the TPD-preferred assignment according to Tab. 1c. Weights are printed for TPD/DPD/SPD, only one number indicates that the weights are equal for all dominance semantics.

	Skier *A*	Boarder *B*	Rookie *C*
$t_X^{(1)}$	**3**	8	7
$t_X^{(2)}$	9	13	16
$t_X^{(3)}$	5	**7**	**5**

(a) SPD semantics

	Skier *A*	Boarder *B*	Rookie *C*
$t_X^{(1)}$	**3**	11	17
$t_X^{(2)}$	14	13	30
$t_X^{(3)}$	10	**10**	**5**

(b) DPD semantics

	Skier *A*	Boarder *B*	Rookie *C*
$t_X^{(1)}$	**3**	27	44
$t_X^{(2)}$	17	**19**	65
$t_X^{(3)}$	13	25	**6**

(c) TPD semantics

Table 1: Different tours rated by different relationship graphs.

4.2 Changing Preferences

In multi-agent-systems, agents change their goals depending on their perceived environmental situation [10]. Similarly, our personas may change their constraint relationships given new circumstances.

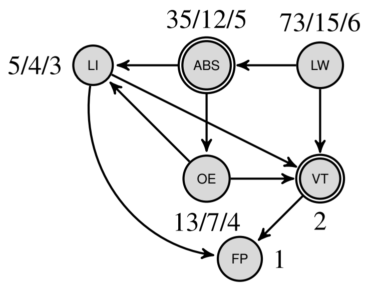
Assume for instance, rookie *C* has gotten enough practice such that avoiding black slopes is not as important as before — but he is not keen on waiting long any more. Hence, the edge $\mathsf{ABS} \succ_R \mathsf{LW}$ gets inverted such that $\mathsf{LW} \succ_R \mathsf{ABS}$, making LW the most important constraint. Assignment $t_X^{(1)}$ is the only one that has a route without much waiting — we expect this as the new favored outcome.

Figure 3 shows that this is indeed the case. One inverted edge led to a revisited weighting that influences the solution preference.

4.3 Changing Constraints

A similar, frequently arising use case is that of a change in available preferences and constraints, especially in open-world scenarios and dynamic CSPs. Assume that slopes and lift endpoints have been evaluated for beautiful landscapes (BL). Additionally, foggy slopes (FS) can be avoided.

Assuming *A* does not care too much about either landscapes or foggy slopes it is safe to assume that those constraints would be ranked even less important than LI. Boarder *B*, however, does care about BL about as



(a) Graph for rookie C new

	Rookie C new
$t_X^{(1)}$	**7**
$t_X^{(2)}$	19
$t_X^{(3)}$	8

(b) SPD semantics

	Rookie C new
$t_X^{(1)}$	**14**
$t_X^{(2)}$	39
$t_X^{(3)}$	17

(c) DPD semantics

	Rookie C new
$t_X^{(1)}$	**35**
$t_X^{(2)}$	127
$t_X^{(3)}$	75

(d) TPD semantics

Fig. 3: After adaption, $t_X^{(1)}$ is now preferred by rookie C

much as variety and marks them more important than FS, FP, and ABS. Assume further that $t_X^{(1)} \models \{\mathsf{BL}, \mathsf{FS}\}$, $t_X^{(2)} \models \{\mathsf{FS}\}$ and $t_X^{(3)} \models \{\mathsf{FS}\}$.

It is easy to calculate that A still ranks $t_X^{(1)} >_R^{\text{TPD}} t_X^{(3)} >_R^{\text{TPD}} t_X^{(2)}$ even though different numeric values are placed. To keep this order while including the new constraints, *six* weights would need to be changed while adding *two* new relationships is enough. For B the situation is different, as BL is only satisfied by $t_X^{(1)}$ which is why it would then be the preferred solution in all dominance properties.

5 Conclusions and Future Work

We have presented constraint relationships, an approach to flexibly and intuitively express preferences over constraints in a qualitative way. Constraint relationships establish a partial preference order on the constraints that can be transformed into a k-weighted CSP while preserving dominance properties. The process is tool-supported and can be applied to static and dynamic problems. Classical solution algorithms for weighted CSPs or off-the-shelf constraint optimizers can be used to solve the resulting problem. We showed a typical example that benefits from constraint relationships.

The approach is especially well-suited for applications in which constraints are added and changed at runtime. As we are mainly concerned with the engineering of self-adaptive systems, we are going to apply the approach to the dynamic adaptation of CSPs that occur in these systems. From a theoretical standpoint, we want to explore different dominance properties and find a minimal c-semiring that respects the rules presented in this paper — but does not introduce any additional ordering relation unlike the k-weighted c-semiring which makes any two solutions comparable. Additionally, we want to examine how constraint relationships can contribute to preference elicitation or preference learning. In particular, elicitation of constraint relationships by comparing solutions satisfying different soft constraints using abductive reasoning is planned.

Currently, we are looking into the use of constraint relationships to synthesize different sub-models into a global CSP in a multi-agent-system. Such a process would allow the combination of different characteristics of individual agents that are not known at design time into a common model that can then be solved centrally. One of the applications of this technique are decentralized energy management systems [23] in which different power plants are combined in a self-organizing fashion to best fulfill a changing power load.

Acknowledgements

This work has been partially funded by the German Research Foundation (DFG) in the research unit FOR 1085 "OC Trust – Trustworthy Organic Computing Systems". We would like to thank María Victoria Cengarle for fruitful discussions and the anonymous reviewers for the constructive feedback that led to an improvement of the manuscript.

References

1. Andréka, H., Ryan, M., Schobbens, P.Y.: Operators and Laws for Combining Preference Relations. J. Log. Comput. **12**(1), 13–53 (2002)
2. Bessière, C.: Arc-Consistency in Dynamic Constraint Satisfaction Problems. In: T.L. Dean, K. McKeown (eds.) Proc. 9th Nat. Conf. Artificial Intelligence (AAAI'91), pp. 221–226. AAAI Press (1991)
3. Bistarelli, S., Montanari, U., Rossi, F.: Constraint Solving over Semirings. In: Proc. 14th Int. Joint Conf. Artificial Intelligence (IJCAI'95), vol. 1, pp. 624–630. Morgan Kaufmann (1995)
4. Borning, A., Freeman-Benson, B., Wilson, M.: Constraint Hierarchies. LISP Symb. Comp. **5**, 223–270 (1992)
5. Boutilier, C., Brafman, R.I., Domshlak, C., Hoos, H.H., Poole, D.: CP-nets: A Tool for Representing and Reasoning with Conditional Ceteris Paribus Preference Statements. J. Artif. Intell. Res. **21**, 135–191 (2004)
6. Boutilier, C., Brafman, R.I., Geib, C., Poole, D.: A Constraint-based Approach to Preference Elicitation and Decision Making. In: AAAI Spring Symp. Qualitative Decision Theory, pp. 19–28 (1997)
7. Brafman, R., Domshlak, C.: Preference Handling – An Introductory Tutorial. AI Magazine **30**(1), 58–86 (2009)
8. IBM ILOG CPLEX Optimizer (2013). `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`
9. Dechter, R., Dechter, A.: Belief Maintenance in Dynamic Constraint Networks. In: H.E. Shrobe, T.M. Mitchell, R.G. Smith (eds.) Proc. 7th Nat. Conf. Artificial Intelligence (AAAI'88), pp. 37–42. AAAI Press (1988)
10. Doyle, J., McGeachie, M.: Exercising Qualitative Control in Autonomous Adaptive Survivable Systems. In: Proc. 2nd Int. Conf. Self-adaptive software: Applications (IWSAS'01), *Lect. Notes Comp. Sci.*, vol. 2614, pp. 158–170. Springer (2003)
11. Dubois, D., Fargier, H., Prade, H.: The Calculus of Fuzzy Restrictions as a Basis for Flexible Constraint Satisfaction. In: Proc. 2nd IEEE Int. Conf. Fuzzy Systems, vol. 2, pp. 1131–1136 (1993)
12. Freuder, E.C., Wallace, R.J.: Partial Constraint Satisfaction. Artif. Intell. **58**(1–3), 21–70 (1992)
13. Gelain, M., Pini, M.S., Rossi, F., Venable, K.B.: Dealing with Incomplete Preferences in Soft Constraint Problems. In: C. Bessière (ed.) Proc. 13th Int. Conf. Principles and Practice of Constraint Programming (CP'07), *Lect. Notes Comp. Sci.*, vol. 4741, pp. 286–300. Springer (2007)
14. Jampel, M.: A Brief Overview of Over-constrained Systems. In: M. Jampel, E. Freuder, M. Maher (eds.) Over-Constrained Systems, *Lect. Notes Comp. Sci.*, vol. 1106, pp. 1–22. Springer (1996)
15. Larrosa, J.: Node and Arc Consistency in Weighted CSP. In: R. Dechter, R.S. Sutton (eds.) Proc. 18th Nat. Conf. Artificial Intelligence (AAAI'02), pp. 48–53. AAAI Press (2002)
16. Larrosa, J., Schiex, T.: Solving Weighted CSP by Maintaining Arc Consistency. Artif. Intell. **159**(1–2), 1–26 (2004)
17. Meseguer, P., Rossi, F., Schiex, T.: Soft Constraints. In: F. Rossi, P. van Beek, T. Walsh (eds.) Handbook of Constraint Programming, chap. 9. Elsevier (2006)
18. Mittal, S., Falkenhainer, B.: Dynamic Constraint Satisfaction. In: H.E. Shrobe, T.G. Dietterich, W.R. Swartout (eds.) Proc. 8th Nat. Conf. Artificial Intelligence (AAAI'90), vol. 1, pp. 25–32. AAAI Press (1990)
19. Rossi, F.: Preferences, Constraints, Uncertainty, and Multi-Agent Scenarios. In: Proc. Int. Symp. Artificial Intelligence and Mathematics (ISAIM'08) (2008)
20. Rossi, F., Venable, K.B., Walsh, T.: Preferences in Constraint Satisfaction and Optimization. AI Mag. **29**(4), 58–68 (2008)
21. Schiex, T., Fargier, H., Verfaillie, G.: Valued Constraint Satisfaction Problems: Hard and Easy Problems. In: Proc. 14th Int. Joint Conf. Artificial Intelligence (IJCAI'95), vol. 1, pp. 631–639. Morgan Kaufmann (1995)
22. Shapiro, L.G., Haralick, R.M.: Structural descriptions and inexact matching. IEEE Trans. Pattern Analysis and Machine Intelligence **PAMI-3**(5), 504–519 (1981)
23. Steghöfer, J.P., Anders, G., Siefert, F., Reif, W.: A System of Systems Approach to the Evolutionary Transformation of Power Management Systems. In: Proc. INFORMATIK 2013 Wsh. "Smart Grids", Lect. Notes Inform. Bonner Köllen Verlag (2013)