Modelling Tradeoffs using Soft Constraints *

Stefano Bistarelli[1,2] and Barry O'Sullivan[3]

[1] Istituto di Informatica e Telematica, CNR, Pisa, Italy
Stefano.Bistarelli@iit.cnr.it
[2] Dipartimento di Scienze
Universitá degli Studi "G. D'annunzio" di Chieti-Pescara, Italy
bista@sci.unich.it

[3] Cork Constraint Computation Centre
Department of Computer Science, University College Cork, Ireland
b.osullivan@cs.ucc.ie

Abstract. Tradeoffs have been proposed in the literature as an approach to resolving over-constrainedness in interactive constraint-based tools, such as product configurators. It has been reported how tradeoffs can be modeled as additional constraints. This paper presents a formal framework for tradeoff generation based on the semiring approach to soft constraints. In particular, user preferences and tradeoffs are, respectively, represented as soft constraints and as an entailment operator. The entailment operator is used to interactively generate new constraints representing tradeoffs. We introduce also a novel definition of substitutability for soft constraints upon which we present a relaxed definition of tradeoffs.

1 Introduction

A typical interactive configuration session is one where a human user articulates preferences for product features to a configurator which ensures consistency between the constraints of the problem and the user's desires. During such a session a point may be reached where all of the user's desires cannot be met. At this point the user could consider "tradeoffs" between his preferences. For example, in configuring a camera, the user may find that it is impossible to have one "*weighting less that 10 ounces with a zoom lens of 10X or more*", but could accept a tradeoff: "*I will increase my weight limit to 14 ounces if I can have a zoom lens of 20X or more.*" Ideally, we would like the configurator to suggest appropriate tradeoffs to the user.

In this paper we extend and formalize previous work on tradeoffs. In [8] tradeoffs are crisp binary constraints that are interactively generated to substitute strict unary crisp constraints representing user desires. In this paper we

* This work has received support from Enterprise Ireland under their Basic Research Grant Scheme (Grant Number SC/02/289) and their International Collaboration Programme (Grant Number IC/2003/88).

increase the utility of tradeoff generation since the amount of information gathered from the user is increased using soft constraints to represent preferences. The ability to capture the preferences of the user in a formal way was not addressed in the earlier work in this area. Furthermore, the extended framework presented here is general enough to deal with any arity of preference constraints, not only unary ones as reported in [8].

The task of tradeoff generation is also formalized in this paper. The possible tradeoffs are given as the result of an "entailment" function. A filter function is used to select one of the possible entailed tradeoff constraints. The final generalization in our framework is that tradeoff constraints are not necessarily limited to being binary. This provides us with a richer model of tradeoff. For example, we may wish to add a non-binary tradeoff constraint in certain situations, such as when we are prepared to have a pair of constraints, c_x and c_y, replaced by a ternary constraint, $c'_{(x,y,z)}$. The additional constraining influence on z could be regarded as an *imposed* constraint.

To handle both of these extensions we use the semiring-based framework [4] that has been shown to be able to represent both crisp and soft constraints in a uniform way. This general framework also gives us the possibility of expressing *approximate* tradeoffs. When it is not possible to find a tradeoff resulting in a solution better than some fixed level of consistency, we can consider the possibility of suggesting to the user solutions that are not worse than a given *degradation* factor. To do this we extend the notion of soft substitutability [2] from domain values to constraints.

Thus, the contributions of this paper are as follows:

1. a formal and general theoretical framework for tradeoff generation for interactive constraint processing that uses soft constraints to represent user preferences and an entailment operator to generate tradeoffs;
2. a notion of substitutability for soft constraints and a relaxed definition of tradeoff based on substitutability and solution degradation.

The remainder of the paper is organized as follows. Section 2 presents the necessary background on the semiring-based approach to handling soft constraints and on the tradeoff generation schema. Section 3 presents our general framework for tradeoff generation. An example is presented in Section 4. An extension to our approach, for computing approximate tradeoffs, is presented in Section 5. Some concluding remarks are made in Section 6.

2 Background: Tradeoffs and Soft Constraints

Product configuration is becoming a well studied design activity which is often modeled and solved as a constraint satisfaction problem [11, 7, 10, 1]. In this paper we present a formal framework for tradeoff generation in interactive constraint processing. In the existing literature on this topic, a tradeoff is a binary constraint which substitutes a pair of unary preference constraints; the tradeoff constraint representing a satisfactory compromise for the user [8]. For example,

consider a set, $U = \{c_1, \ldots, c_k\}$, of user-specified unary preference constraints, and a set P of constraints defining the underlying problem, such that $U \cup P$ is inconsistent. A tradeoff constraint, T_{ij}, is a binary constraint involving a pair of variables, v_i and v_j, on which the user has specified a pair of unary preference constraints, $c_i \in U$ and $c_j \in U$, such that $U \cup T_{ij} - \{c_i, c_j\}$ is consistent and the user's preference on one variable has been strengthened and relaxed on the other. Therefore, currently, a tradeoff is a binary constraint which replaces a pair of unary preference constraints defined over the same pair of variables.

In this paper we regard each constraint in U as a *soft constraint* whose preference levels can be combined accordingly to the specific notion of combination for the problem. Soft constraints associate a qualitative or quantitative value either to the entire constraint or to each assignment of its variables. Such values are interpreted as a level of preference, importance or cost. The levels are usually ordered, reflecting the fact that some levels (constraints) are *better* than others. When using soft constraints it is necessary to specify, via suitable combination operators, how the level of preference of a global solution is obtained from the preferences in the constraints.

Several formalizations of the concept of *soft constraints* are currently available. In the following, we refer to the formalization based on c-semirings [4, 6], which can be shown to generalize and express both crisp and soft constraints [3].

A semiring-based constraint assigns to each instantiation of its variables an associated value from a partially ordered set. When dealing with crisp constraint, the values are the booleans *true* and *false* representing the admissible and non-admissible values; when dealing with soft constraints the values are interpreted as preferences.

The framework must also handle the combination of constraints. To do this one must take into account such additional values, and thus the formalism must provide suitable operations for combination ($\times$) and comparison ($+$) of tuples of values and constraints. This is why this formalization is based on the concept of c-semiring. Below we present an overview of semiring-based constraint satisfaction.

Semirings. A semiring is a tuple $\langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ such that:

- A is a set and $\mathbf{0}, \mathbf{1} \in A$;
- $+$ is commutative, associative and $\mathbf{0}$ is its unit element;
- $\times$ is associative, distributes over $+$, $\mathbf{1}$ is its unit element and $\mathbf{0}$ is its absorbing element.

A c-semiring is a semiring $\langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ such that: $+$ is idempotent, $\mathbf{1}$ is its absorbing element and $\times$ is commutative. Let us consider the relation $\leq_S$ over A such that $a \leq_S b$ iff $a + b = b$. Then it is possible to prove that (see [4]):

- $\leq_S$ is a partial order;
- $+$ and $\times$ are monotone on $\leq_S$;
- $\mathbf{0}$ is its minimum and $\mathbf{1}$ its maximum;
- $\langle A, \leq_S \rangle$ is a complete lattice and, for all $a, b \in A$, $a + b = lub(a, b)$.

Moreover, if $\times$ is idempotent, then: $+$ distribute over $\times$; $\langle A, \leq_S \rangle$ is a complete distributive lattice and $\times$ its glb. Informally, the relation $\leq_S$ gives us a way to compare semiring values and constraints. In fact, when we have $a \leq_S b$, we will say that b *is better than* a. In the following, when the semiring will be clear from the context, $a \leq_S b$ will be often indicated by $a \leq b$.

Constraint Problems. Given a semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ and an ordered set of variables V over a finite domain D, a *constraint* is a function which, given an assignment $\eta : V \to D$ of the variables, returns a value of the semiring.

By using this notation we define $\mathcal{C} = \eta \to A$ as the set of all possible constraints that can be built starting from S, D and V.

Note that in this *functional* formulation, each constraint is a function (as defined in [6]) and not a pair (as defined in [4]). Such a function involves all the variables in V, but it depends on the assignment of only a finite subset of them. We call this subset the *support* of the constraint.

Consider a constraint $c \in \mathcal{C}$. We define his support as $supp(c) = \{v \in V \mid \exists \eta, d_1, d_2 . c\eta[v := d_1] \neq c\eta[v := d_2]\}$, where

$$\eta[v := d]v' = \begin{cases} d & \text{if } v = v', \\ \eta v' & \text{otherwise.} \end{cases}$$

Note that $c\eta[v := d_1]$ means $c\eta'$ where η' is η modified with the association $v := d_1$ (that is the operator $[\ ]$ has precedence over application).

A *soft constraint satisfaction problem* is a pair $\langle C, con \rangle$ where $con \subseteq V$ and C is a set of constraints: *con* is the set of variables of interest for the constraint set C, which however may concern also variables not in *con*.

Note that a classical CSP is a SCSP where the chosen c-semiring is: $S_{CSP} = \langle \{false, true\}, \vee, \wedge, false, true \rangle$. Fuzzy CSPs can instead be modeled in the SCSP framework by choosing the c-semiring $S_{FCSP} = \langle [0, 1], max, min, 0, 1 \rangle$. Many other "soft" CSPs (Probabilistic, weighted, ...) can be modeled by using a suitable semiring structure, such as, ($S_{prob} = \langle [0, 1], max, \times, 0, 1 \rangle$, $S_{weight} = \langle \mathcal{R}, min, +, 0, +\infty \rangle$, ...).

Combining constraints. When there is a set of soft constraints $\mathcal{C}$, the combined weight of the constraints is computed using the operator $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ defined as $(c_1 \otimes c_2)\eta = c_1\eta \times_S c_2\eta$.

Given a constraint $c \in \mathcal{C}$ and a variable $v \in V$, the *projection* of c over $V - \{v\}$, written $c \Downarrow_{(V-\{v\})}$ is the constraint c' s.t. $c'\eta = \sum_{d \in D} c\eta[v := d]$. Informally, projecting means eliminating some variables from the support. This is done by associating to each tuple over the remaining variables a semiring element which is the sum of the elements associated by the original constraint to all the extensions of this tuple over the eliminated variables. In short, combination is performed via the multiplicative operation of the semiring, and projection via the additive one.

Solutions. The *solution* of a SCSP $P = \langle C, con \rangle$ is the constraint $Sol(P) = (\bigotimes C) \Downarrow_{con}$. That is, we combine all constraints, and then project over the variables in *con*. In this way we get the constraint with support (not greater than) *con* which is "induced" by the entire SCSP. Note that when all the variables are of interest we do not need to perform any projection.

Solutions are constraints in themselves and can be ordered by extending the $\leq_S$ order. We say that a constraint c_1 is at least as constraining as constraint c_2 if $c_1 \sqsubseteq c_2$, where for any assignment η of variables then $c_1 \sqsubseteq c_2 \equiv c_1\eta \leq_S c_2\eta$.

Sometimes it may be useful to find only a semiring value representing the least upper bound among the values yielded by the solutions. This is called the *best level of consistency* of an SCSP P and it is defined by $blevel(P) = Sol(P) \Downarrow_{\emptyset}$. We say that P is α-consistent when $blevel(P) = \alpha$.

3 Tradeoff as an Entailment Operator

In this section we define a general notion of tradeoff using the semiring-based framework presented above. We use hard constraints to represent the strict and unmodifiable conditions of the problem (represented by P). We use soft constraints to represent, modifiable, user preferences (represented as U).

While, in general, we may have some constraints in P which may be soft, we would regard this softness as a cost that would not be handled in the same way as user-specified preference constraints. In our framework we model softness in the users desires, U, as preferences. The user makes statements like *"I prefer to have a petrol engine over a diesel one"* in a quantifiable manner by associating semiring values which each option. However, any softness in the physical constraints in the problem, P, represent the costs, or penalties, associated with relaxing them. These costs can be thought of as problem statements such as *"a diesel engine is not normally available for the small chassis, but for an additional cost, we can make the option available"*. Note that these types of softness are semantically different and we would treat them as such. For the remainder of the paper we will simply regard each problem constraints in P as a hard (crisp) constraint.

We model tradeoffs as a special *entailment operator* [12]. As shown in [6], an entailment operator for soft constraints, given a set of constraints C, generates constraints c' s.t. $\bigotimes C \sqsubseteq c'$ (written as $C \vdash c'$).

Tradeoffs specialize entailments in two respects:

- firstly, constraints c' generated by the tradeoff operator are *substituted* for the preference constraints, C, while entailed constraints are usually *added* to the problem.
- secondly, when we add tradeoffs, we do not necessarily obtain a globally better solution, but we may obtain a *Pareto optimal* one. Specifically, while low preference constraints, $C \in U$, are substituted by c', thus increasing the overall level of preference, c' usually also lowers the preference of some other constraints $\bar{C} \in U$.

So, for instance, if $U = \{c_1, \ldots, c_n\}$, a tradeoff for constraints $C = \{c_1, c_2\}$ could be a constraint c' s.t. $C \vdash c'$ and $supp(c') \supseteq supp(c_1) \cup supp(c_2)$ (usually we have $supp(c') \supset supp(c_1) \cup supp(c_2)$). Formally, we can define the notion of potential tradeoffs as follows:

Definition 1 (Potential Tradeoffs). *Given a configuration problem $\{P \cup U\}$, and a subset of user preference constraints $C \subseteq U$. We say that c' is a* Potential Tradeoff *for C ($c' \in Trades_{\langle P,U \rangle}(C)$) if*

- $supp(\bigotimes C) \subseteq supp(c')$*; let's call $\bar{C} \subseteq U$ the greatest set of preference constraints s.t. $supp(\bigotimes\{C, \bar{C}\}) = supp(c')$;*
- $C \vdash c' \Downarrow_{supp(C)}$*;*
- $\bigotimes\{P \cup \{U - C\} \cup c'\} \not\sqsubseteq \bigotimes\{P \cup U\}$*;*

The meaning of this definition is that a potential tradeoff will increase the level of some user preference constraints (those in $C = \{c_1, c_2\}$), and possibly lower some other ones whose support is in $supp(c')/(supp(c_1) \cup supp(c_2))$ (let call $\bar{C}$ this set of constraints: $\bar{C} = \{c_i : supp(c_i) \subseteq supp(c')/(supp(c_1) \cup supp(c_2))\}$). Notice that after the application of the tradeoff operator, we never obtain a solution worse than before.

3.1 Computing Tradeoffs

The way the preference constraints $C \subseteq U$ are selected, and the way a specific tradeoff c' is filtered from all the potential ones, and thus, $\bar{C}$ computed, is one of the most important issues when dealing with configuration problems. The potential tradeoffs can be restricted in various ways, which may be problem or context-specific. For example, we may wish to select a tradeoff in a way which could be regarded as "user-friendly". In the context of an interactive constraint-based configuration tool we may need to ensure that the user "trusts" the configurator, which may imply that previously accepted tradeoffs are not revisited. Some possible approaches to selecting preferences and filtering tradeoff constraints are presented in Section 3.2.

Notice that the filtered tradeoff could depend on the presence of a particular (partial) assignment, η, of the variables of the problem whose association has to be maintained[1].

Usually the configuration process first requests preference constraints from the user, and only if a solution better than a threshold α cannot be found with the acquired constraints, then a tradeoff is computed. To perform this check we

[1] In this paper η is fixed and never changes, so in all the definitions it could be omitted. Nevertheless it is important to include it in the notion of preference/tradeoff because in the next step of a configuration problem it will play an important role. After giving the user the opportunity to specify constraint preferences, he will be asked to make some more precise choices. In that phase of the configuration process η becomes a partial assignment. We plan to address this second phase of the configuration problem as part of our research agenda in this area.

always compute $blevel(P \cup U) = Sol(P \cup U) \Downarrow_{\emptyset}$ and compare this value with the semiring level α.

Therefore, tradeoff generation can be characterized in this context as a function:

$$Trades^{\alpha}_{\langle P,U,\eta,! \rangle} : \mathcal{C} \rightarrow \mathcal{C}$$

where:

- P is the set of *Problem constraints*;
- U is the set of *User-specified preference constraints*;
- ! is a filter (cut function) that first selects a set of preference constraints, $C \subseteq U$, and then select a possible tradeoff constraint c' (see Section 3.2).
- η is a (partial) assignment of the variables whose association has to be maintained;
- α represents the minimum best level of consistency we wish to achieve in the SCSP. The level α can be seen as the minimum level of global preference satisfaction we want to achieve;

Definition 2 (Tradeoffs). *Given a configuration problem* $\{P \cup U\}$*, and a subset of user preference constraints* $C \subseteq U$*. We say that* c' *is a* Tradeoff *for* C *using the threshold* α*, the filter* ! *and the partial assignment* η *(*$c' \in Trades^{\alpha}_{\langle P,U,\eta,! \rangle}(C)$*), if the following are satisfied:*

- c' *is a* Potential Tradeoff*, that is:*
  - $supp(\bigotimes C) \subseteq supp(c')$*;*
  - $C \vdash c' \Downarrow_{supp(C)}$*;*
  - $\bigotimes\{P \cup \{U - C\} \cup c'\} \not\sqsubseteq \bigotimes\{P \cup U\}$*;*
- $blevel(\{P \cup U\}) < \alpha \leq blevel(\{P \cup \{U - C\} \cup c'\})$ *(starting from a solution with an insufficient level of consistency, we want an assignment that gives a solution with a best level of consistency of at least* α[2]*).*
- ! *is used as a filter (see Section 3.2).*

3.2 Heuristics for selecting preference and tradeoff constraints

To completely define a tradeoff function we need to specify the selection and filtering heuristic, !, we will use to:

1. select the user preference constraints to eliminate C – we will denote the selector $!^{out}$), and
2. select a tradeoff c' from the set of potential tradeoffs computed by the entailment operator – we will denote the filter $!^{in}$; notice that this also implies the selection of the preference constraints $\bar{C}$ whose level of preference could be reduced.

[2] We will relax this condition by using a notion of degradation in Section 5.

Before presenting some specific examples of the selection heuristic, $!^{out}$, and the filtering heuristic, $!^{in}$, recall that the trigger for generating tradeoff constraints is the detection that there does not exist a solution which is α-consistent, i.e. that $blevel(\{P \cup U\}) < \alpha$. Below, we give here some possible instantiations of $!^{out}$ and $!^{in}$.

Random Selection: Random selection is always a possibility. In this case, the set C containing the preference constraint(s) to remove is randomly selected among those in $!^{out}(U) = \{c_j \in U : P \cup U - \{c_j\}$ is α-consistent$\}$; This heuristic is an obvious naive approach.

Strictly related to P: The preference constraint we want to modify is strictly connected to the problem definition. In this case, the set C containing the preference constraint(s) to remove is randomly selected among those in $!^{out}(U) = \{c_j \in U : P \cup U - \{c_j\}$ is α-consistent $\wedge\ \exists c_i \in P : supp(c_i) \cap supp(c_j) \neq \emptyset\}$;

Has not appeared in a tradeoff: The preference constraint we want to modify has not already been affected by a tradeoff. In this case, the set C containing the preference constraint(s) to remove is randomly selected among those in $!^{out}(U) = \{c_j \in U : P \cup U - \{c_j\}$ is α-consistent $\wedge\ \nexists$ a tradeoff $t \in U : supp(t) \supseteq supp(c_j)\}$; this heuristic can be regarded as "user-friendly" since we do not ask the user to consider tradeoffs on variables which have already been involved in a tradeoff.

The tradeoff constraint, c', that we will choose, by filtering using $!^{in}$, has the properties that it will: (a) reflect a relaxation of the user's preference for constraint(s) C (selected by using $!^{out}$, and (b) a strengthening of the user's preference for constraint(s) $\bar{C}$. By relaxation (resp. strengthening) of a user's preference we mean that when we project the constraint c' onto $supp(C)$ (resp. $supp(\bar{C})$), it will be less (resp. more) restrictive than C (resp. $\bar{C}$).

Proposition 1. *Given a configuration problem $\{P \cup U\}$, a subset of user preference constraints $C \subseteq U$ and a tradeoff c' for C; let also $\bar{C} \subseteq U$ the greatest set s.t. $supp(\bigotimes\{C, \bar{C}\}) = supp(c')$; Then,*

- $(P \cup \{U - C\} \cup c') \Downarrow_{supp(C)} \sqsupseteq (P \cup U) \Downarrow_{supp(C)}$;
- $(P \cup \{U - C\} \cup c') \Downarrow_{supp(\bar{C})} \sqsubseteq (P \cup U) \Downarrow_{supp(\bar{C})}$.

Let's give now some example of $!^{in}$ filters that among all possible potential tradeoff can select only one. Some possible approaches we could adopt, based on [8] are:

Maximum viability *c' is maximal w.r.t. P, U and η (that is for all $c'' \in Trades^{\alpha}_{\langle P,U,\eta,! \rangle}$ we have $\bigotimes\{P \cup \{U - C\} \cup c''\}\eta \not\geq \bigotimes\{P \cup \{U - C\} \cup c'\}\eta$;*

Minimum viability *c' is minimal w.r.t. P, U and η (that is there not exists $c'' \in Trades^{\alpha}_{\langle P,U,\eta,! \rangle}$ s.t. $\bigotimes\{P \cup \{U - C\} \cup c''\}\eta \leq \bigotimes\{P \cup \{U - C\} \cup c'\}\eta$.*

Notice that the first approach will be less tasking on the configurator since it selects always the less restrictive tradeoff, i.e. we will have $c' \Downarrow_{supp(C)} = \mathbf{1}$; the tradeoff will give to all domain values of the variables in C the best preference. In this way the several preferences made by the user on this assignment are lost.

On the other hand, the second approach will try to always stay as close as possible to the user's preferences, i.e. we will have $c' \Downarrow_{supp(C)} \sqsupset C$. The tradeoff will increase the preference on C just sufficiently to reach the prefixed level of consistency α. Therefore, such a minor increment could result in a significant number of tradeoff interactions during the configuration process. In fact, the constraint c' inserted by the configurator could be too strict to be α-consistent when the user will insert new preference constraints in future interactions.

It is worth pointing out at this point that a good user-interface could assist in the detection of preference constraints to restrict and which to relax. An appropriate user-interface could also take care of preference elicitation. For example, we could assume that importance relationships between variables is reflected by the order in which user-choices are made. This is also an issue we are investigating as part of our research agenda in this area.

4 An Example

The example configuration problem that will be studied here is based on the soft n-queens problem [5]. However, we will assume, for space reasons, that all of the problem constraints in P are hard in the traditional sense. Furthermore, for this example we have chosen the fuzzy semiring, $S = \langle [0,1], max, min, 0, 1 \rangle$, so constraints in U have preference between 0 and 1 and they are combined using min.

The user attempts to solve this configuration problem by interactively specifying some preference constraints to a constraint-based configurator. During the interactive session with the configurator, the user may specify a preference constraint which causes the problem to become "over-constrained", identified by the problem becoming less than α-consistent (for some fixed α). At this point our configurator attempts to recommend tradeoff constraints to the user which he/she can accept before continuing.

Thus, the interactive solving process, based on [8], can be summarized as follows:

Repeat until preferences are specified for all variables in the problem:
- *Repeat until over-constrained – $blevel(P \cup U) < \alpha$:*
  - *the user specifies a preference constraint, $c \in U$;*
- *Repeat until user is satisfied:*
  - *the system proposes a tradeoff*

For the purposes of this example lets assume we wish to solve the 4-Queens problem with a crisp set of problem constraints, P, i.e. the configuration problem constraints are hard. This means that the only preference/costs we have to consider are given by the user (inside set U).

In the following example we will consider only unary preference constraints representing the columns, $\{c_1, c_2, c_3, c_4\}$ of the chess-board. The user proposes unary preference constraints on each of the columns, sequentially from the first to the fourth, representing wishes for the placement of queens.

Furthermore, we also assume that only binary tradeoff constraints c' will be generated; Finally, lets assume that the user wishes to achieve a minimum level of 0.5-consistency, i.e. $\alpha = 0.5$.

Problem constraints in P: Figure 1(a) the *no-attack* crisp constraints in$P$ are represented. The gray square represent impossible configuration (that is with preference level 0); the white square represent instead possible position for the queens.

User Decision #1: The user states a unary preference constraint on the values for column 1: $C_1(1) = 1.0, C_1(2) = 0.6, C_1(3) = 0.6, C_1(4) = 1.0$ (see Figure 1(b)). The constraint network representing this problem is still α-consistent, because it is possible to achieve a solution having a semiring value of 0.6 (Figure 1(c)) by putting the queen in row 2 or 3, so the user is free to continue articulating preference constraints. Notice that queens cannot be positioned in row 1 or row 4 because the crisp constraints in P do not admit any solution with queens in these positions.

User Decision #2: The user states a unary preference constraint on the values for column 2: $C_2(1) = 1.0, C_2(2) = 0.4, C_2(3) = 0.4, C_2(4) = 1.0$ (Figure 1(d)). The constraint network representing this problem is still α-consistent: setting $c_1 = 2$ and $c_2 = 4$ could yield a solution with preference 0.6 (Figure 1(e)). In fact, possible solutions are obtained by putting the queen in row 1 or 4[3] Therefore, the user is free to continue articulating preferences.

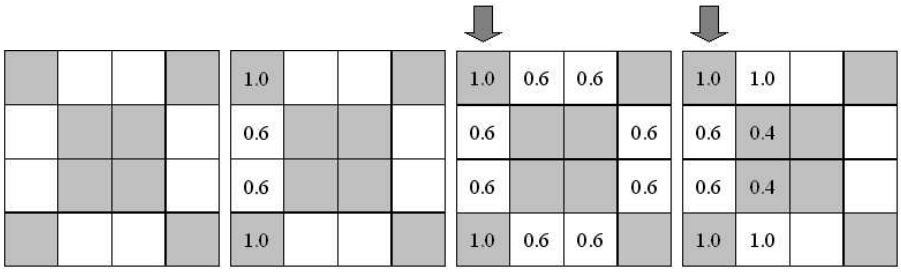
User Decision #3: The user states a unary preference constraint on the values for column 3: $c_3(1) = 0.4, c_3(2) = 1.0, c_3(3) = 1.0, c_3(4) = 0.4$ (Figure 1(f)). The constraint network representing this problem is no longer α-consistent: the best solution has a semiring value of 0.4 (for example setting $c_1 = 2$, $c_2 = 4$ and $c_3 = 1$) (Figure 1(g)). Therefore, the user must consider tradeoffs.

Tradeoff #1: The configurator need to select a preference constraint to remove and a tradeoff to add. In this case all the heuristic presented in Section 3.2 for $!^{out}$ will select the preference constraint on column 3. In fact $C = \{c_3\}$ is the only set of preference constraint s.t. $P \cup U - C$ is 0.5-consistent.

Furthermore, lets assume that the tradeoff constraint selected using $!^{in}$ involve column 1 and 3. Let's suppose our heuristic select: $C_{(1,3)} = \{(4,4) = 1.0, (4,3) = 1.0, (4,2) = 1.0, (4,1) = 1.0, (3,4) = 0.5, (3,3) = 1.0, (3,2) = 1.0, (3,1) = 0.5, (2,4) = 0.5, (2,3) = 1.0, (2,2) = 1.0, (2,1) = 0.5, (1,4) = 1.0, (1,3) = 1.0, (1,2) = 1.0, (1,1) = 1.0\}$ (Figure 1(h)).

Assuming the user accepts this tradeoff constraint, the network is once again α-consistent (Figure 1(i)). The user could set $c_1 = 2$, $c_2 = 4$ and $c_3 = 1$ for a semi-ring value of 0.5. Note that adding the tradeoff we also remove the

[3] In this example all of the constraints in P are crisp, so the possible positions of the queens are strictly imposed. Instead, if P contained soft constraints, more possibilities (with different costs) would be possible.

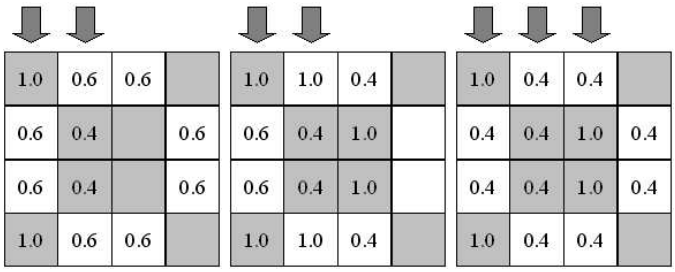


(a) Chessboard with crisp constraints.

(b) Preferences for the 1st column.

(c) Evaluation of the 1st preference.

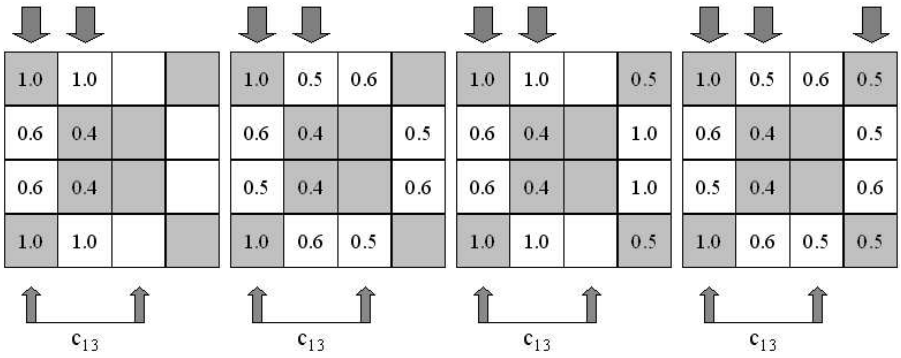
(d) Added preferences for the 2nd column.



(e) Evaluation of the preference for the 1st and 2nd columns.

(f) Added preferences for the 3rd column.

(g) Evaluation of the preferences for 1st,2nd and 3rd columns.



(h) Added a tradeoff between preferences on the 1st and 3rd column.

(i) Evaluation of the preferences for 1st and 2nd columns and of the tradeoff.

(j) Added preferences for the 4th column.

(k) Evaluation after the preferences over the 4th column.

Fig. 1. An example interaction with all hard constraints in P.

unary preference constraint on column 3 and relax the preferences on column 1.

User Decision #4: The user states a unary preference constraint on the values for column 4: $C_4(1) = 0.5, C_4(2) = 1.0, C_4(3) = 1.0, C_4(4) = 0.5$ (Figure 1(j)). The constraint network representing this problem is still α-consistent (Figure 1(k)). The user could set $c_1 = 2$, $c_2 = 4$, $c_3 = 1$ and $c_4 = 3$ for a semi-ring value of 0.6. Preferences have now been defined for each column.

5 Relaxing Tradeoffs

Sometimes a tradeoff that completely satisfies the user's desires does not exist. This results in a dead-end being reached, where the configurator cannot assist the user to achieve the desired level of consistency. One possibility to resolve this situation is to assist the user in backing-up through the tradeoffs that the user accepted and modify one or more of them. Alternatively, a notion of less strict tradeoff is useful. In this Section we define a relaxed version for tradeoffs based on a notion of substitutability for soft constraints which extends a definition of substitutability among domain values [2]. In Section 5.1 we present a generalization of the notion of soft value substitutability to substitutability for soft constraints. We then use this definition in Section 5.2 to relax the earlier notion of tradeoff.

5.1 Substitutability for Soft Constraints

We propose a definition of substitutability among constraints, extending the notions of substitutability among values defined in [2]. Informally, according to the definition of soft value substitutability, a domain value b is substitutable for a domain value a for a variable v if all solutions involving $v := b$ have a level of preference greater than solutions involving $v := a$. We extend this definition to constraints below.

Definition 3 (constraint substitutability/interchangeability). *We say that constraint c_1 is Fully Substitutable for constraint c_2 ($c_1 \in FS(c_2)$) w.r.t. the set of constraints C and a (partial) assignment η when $\bigotimes\{C \cup c_2\}\eta \sqsubseteq \bigotimes\{C \cup c_1\}\eta$. When we have $\bigotimes\{C \cup c_2\}\eta = \bigotimes\{C \cup c_1\}\eta$ we say that c_1 and c_2 are Fully Interchangeable ($FI(c_1/c_2)$).*

Note that this definition of substitutability for constraints, when dealing with crisp constraints, is very similar to that defined in [9]. However, the definition above is more general since we are dealing with the soft constraint framework.

As reported in [2] for domain values, the above definition can be relaxed by using degradations. In particular, a constraint $c*$ can be used as a degradation factor.

Definition 4 (Substitutability with degradation). *Consider two constraints c_1 and c_2, the set of constraints C, a partial assignment η and a* degradation

constraint $c*$; we say that c_1 is ^{c*}Substitutable *for c_2 w.r.t. C and η if and only if,*

$$\bigotimes\{C \cup c_2 \cup c*\}\eta \sqsubseteq \bigotimes\{C \cup c_1\}\eta$$

5.2 A Relaxed Definition of Tradeoffs

The above notion of degradation can be used to relax the tradeoffs that the configurator proposes to the user. In particular, we will define a notion of *approximate* tradeoff. However, first we will show that tradeoffs and substitutability constraints are strictly related.

Theorem 1 (Substitutability and Tradeoffs). *Consider the sets of constraints P, U, and $C \subseteq U$, a threshold α and a partial assignment of the variable η. Then, if c' is substitutable for $\bigotimes C$ ($c' \in FS(\bigotimes C)$) w.r.t. the set of constraints $\{P \cup U\}$ and the (partial) assignment η, then $c' \in Trades^{\alpha}_{\langle P,U,\eta\rangle}(C)$.*

We can now relax the definition of tradeoff by using the relaxed version of substitutability.

Definition 5 (Degradation Tradeoffs). *Consider the constraint c', the (set of) constraint $C \in U$, a set of constraints $\{P \cup U\}$, a partial assignment η and a* degradation *constraint $c*$; we say that c' is a tradeoff for C w.r.t. $\{P \cup U\}$ and η with degradation $c*$ ($c' \in {}^{c*}Trades^{\alpha}_{\langle P,U,\eta\rangle}(C)$) if and only if, c' is* ^{c*}Substitutable *for $\bigotimes C$ w.r.t. $\{P \cup U\}$ and η.*

Degradation tradeoffs can be regarded as *approximate* tradeoffs. Notice that by using the notion of degradation, we do not impose the requirement that the configurator find a better configuration, but rather a configuration *similar* to that chosen by the user. In this way the solution that the configurator will find will be not too far from the solution proposed by the user in terms of its degree of preference. In this respect the solution is an approximation.

5.3 An Example: Relaxing the n-queens

Let's give an easy example of relaxed tradeoffs by using the same example depicted in Section 4. Let's fix $\alpha = 0.5$ as before, but also consider the degradation factor $c* = 0.7$. This means that we want to obtain solutions better than 0.5 if possible, but a degradation to 0.7 would be acceptable[4].

Let's suppose the user is articulating the same choices as before. Until User Decision #3, everything runs as before. However, let's consider what happens after that.

User Decision #3: The user states a unary preference constraint on the values for column 3, and the problem is no longer α-consistent: the best solution has a semiring value of 0.4 (for example setting $c_1 = 2$, $c_2 = 4$ and $c_3 = 1$) (Figure 1(g)). Therefore, the user must consider tradeoffs.

[4] Note that the notion of degradation assumes greater importance when dealing with a non-idempotent semiring. In this case, it really represents the amount of loss we are prepared to accept.

Tradeoff #1: The configurator must select a preference constraint to remove and a tradeoff to add. However, here we consider tradeoffs with degradation 0.7. Let's suppose also in this case to select using $!^{out}$ the set $C = \{c_3\}$. Applying the definition, we want to find the tradeoff c' s.t.

$$\bigotimes\{(P \cup \{c_1, c_2\}) \cup c_3 \cup c*\}\eta \sqsubseteq \bigotimes\{(P \cup \{c_1, c_2\}) \cup c'\}\eta.$$

We can select several c'. Easily we can take $c' \sqsubseteq c_3 \otimes c*$. We have $c_3 \otimes c* = \bar{c}$ s.t. $\bar{c}(1) = min(0.4, 0.7), \bar{c}(2) = min(1.0, 0.7), \bar{c}(3) = min(1.0, 0.7), \bar{c}(4) = min(0.4, 0.7) = \bar{c}(1) = 0.4, \bar{c}(2) = 0.7, \bar{c}(3) = 0.7, \bar{c}(4) = 0.4$. Notice that using the notion of relaxed tradeoff, the new constraint c' does not strictly increase the preferences for the column #3.
Lets now assume that the tradeoff constraint selected using $!^{in}$ involves columns 1 and 3 as before. A possible c' could be the following (the same as before, but, for instance, with the underlined assignment changed): $C_{(1,3)} = \{(4,4) = 1.0, \underline{(4,3) = 0.7}$,
$(4,2) = 1, (4, \overline{1) = 1.0, (3}, 4) = 0.5, \underline{(3,3) = 0.8}, (3,2) = 1.0, (3,1) = 0.5$,
$(2,4) = 0.5, \underline{(2,3) = 0.8}, (2,2) = 1.0, \overline{(2,1) = 0}.5, (1,4) = 1.0, \underline{(1,3) = 0.7}$,
$(1,2) = 1.0, \overline{(1,1) = 1.0}\}$ (Figure 1(h)).
The point is that the added constraint could be worse than before (but not by more that the degradation factor). Moreover the added binary constraint representing the tradeoff, when restricted to variable c_3 (solumn 3) is closer to the original constraint articulated by the user. Assuming the user accepts this tradeoff constraint, the network is once again α-consistent (we obtain the same solution represented in Figure 1(i)). The user can then complete the interaction setting the preference for the last constraint.

Notice that in this example we are dealing with a semiring with an idempotent times operator (the fuzzy one). In this case the notion of degradation for tradeoff is not so evident. Let's suppose to deal instead with a non idempotent times, for instance with a min-sum semiring over natural numbers. In this case we want to minimize the sum of the cost of each constraint. In this case, speaking for instance of a tradeoff of 10 units, means that when we compute a tradeoff for a constraint (c_3 in the example) we can "afford" to loose a maximum of 10 unit w.r.t. our original goal α. So the generated constraint could be also greater (because we minimize) than α but not more by than 10 units.

6 Conclusions and Future Work

Tradeoffs have been proposed in the literature as an approach to resolving over-constrainedness in interactive constraint-based tools, such as product configurators. It has already been reported in the literature how tradeoffs can be modeled as additional constraints. This paper presents a formal framework for tradeoff generation based on the semiring approach to handling soft constraints. In particular, we present a formal and general definition of tradeoff generation for interactive constraint processing. We present a novel definition of substitutability for soft constraints upon which we present a relaxed definition of tradeoffs.

Our research agenda in this area involves studying intelligent interfaces for reasoning about the relative importance of the user's preferences. For example, we could assume that importance relationships between variables is reflected by the order in which user-choices are made. We are also working on an empirical evaluation of a number of heuristics for selecting preference constraints to be considered as the basis for generating tradeoffs and strategies for filtering from the set of tradeoffs generated by our entailment operator.

In summary, we have presented a formal framework for studying a very important aspect of interactive constraint processing, the ability to assist users achieve their desires to the maximal degree possible. This framework provides the basis for a research agenda in the are of interactive constraint satisfaction with practical applications in domains such as product configuration, e-commerce, interactive scheduling, negotiation and explanation.

References

[1] J. Amilhastre, H. Fargier, and P. Marguis. Consistency restoration and explanations in dynamic csps – application to configuration. *Artificial Intelligence*, 135:199–234, 2002.

[2] S. Bistarelli, B. Faltings, and N. Neagu. A definition of interchangeability for soft CSPs. In Barry O'Sullivan, editor, *Recent Advances in Constraints*, volume 2627 of *LNAI*, pages 31–46, 2003.

[3] S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, and G. Verfaillie. Semiring-based CSPs and Valued CSPs: Frameworks, properties, and comparison. *Constraints*, 4(3), 1999.

[4] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based Constraint Solving and Optimization. *Journal of the ACM*, 44(2):201–236, Mar 1997.

[5] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based Constraint Logic Programming: Syntax and Semantics. *ACM Transactions on Programming Languages and System (TOPLAS)*, 23:1–29, 2001.

[6] S. Bistarelli, U. Montanari, and F. Rossi. Soft concurrent constraint programming. In *Proc. ESOP, Grenoble, France*, number 2305 in LNCS. Springer-Verlag, 2002.

[7] A. Felfernig, G. Friedrich, D. Jannach, and M. Stumpter. Consistency-based diagnosis of configuration knowledge-bases. In *Proceedings of the 14h European Conference on Artificial Intelligence (ECAI'2000)*, pages 146–150, 2000.

[8] E. C. Freuder and B. O'Sullivan. Generating tradeoffs for interative constraint-based configuration. In *Proceedings of CP-2001*, pages 590–594, Nov 2001.

[9] P. Jeavons, D. Cohen, and M. Copper. A substitution operation for constraints. In *Proceedings of the Second International Workshop on Principles and Practice of Constraint Programming – CP-94*, volume 874 of *LNCS*, 1994.

[10] M. Moretti, F. Rossi, E.C. Freuder, C. Likitvivatanavong, and R. Wallace. Explanations and optimisation in preference-based configurators. In Barry O'Sullivan, editor, *Recent Advances in Constraints*, volume 2627 of *LNAI*, pages 76–92, 2003.

[11] D. Sabin and R. Weigel. Product configuration frameworks – a survey. *IEEE Intelligent Systems and their applications*, 13(4):42–49, July–August 1998.

[12] V.A. Saraswat. *Concurrent Constraint Programming.* MIT Press, 1993.