Tight and Tractable Reformulations for Uncertain CSPs

Neil Yorke-Smith* and Carmen Gervet

IC–Parc, Imperial College London, SW7 2AZ, U.K.
{nys,cg6}@icparc.ic.ac.uk

Abstract Various extensions of the CSP framework exist to address ill-defined, real-world optimisation problems. One extension, the *uncertain CSP* (UCSP) tackles the aspect of data errors and incompleteness by ensuring that the problem is faithfully represented with what is known for sure about the data, and by seeking reliable solutions that do not approximate such uncertainties. The extended model has a great impact on the solving complexity. For instance, by introducing bounded interval coefficients, the default representation of an arithmetic linear constraint is of degree 2. A challenge lies in determining constraint classes that allow one to reformulate the UCSP model such that polynomial algorithms exist. In this paper we present two novel sufficient conditions, built on algebraic properties of constraints, that ensure a tractable reformulation exists. We give an algorithm to test for the conditions for binary constraints, and demonstrate as instances some previously identified practical UCSP reformulations.

1 Introduction

Constraint Programming (CP) has proved an effective paradigm to model and solve large scale combinatorial optimisation (LSCO) problems from disparate domains. To date, however, there has been little work on accommodating data uncertainty within CP, apart from seeking a solution that holds in the greatest number of circumstances.

We address the task of modelling and solving uncertain problems where data uncertainty arises due to incomplete or erroneous measurements. It raises the issue of faithfully representing the problem and deriving reliable results that do not approximate the data measurements, but rather take them into account during computation. This form of data uncertainty is ubiquitous in many application domains (e.g. camera control [5], engineering design [3], network inference [20], robot motion [9]). A common CP approach consists of approximating such uncertainties to reach a satisfiable deterministic model, by selecting representative values for the data (e.g. mean values) or using some preliminary error-correction methods. However, the derived model would not ensure a faithful representation of the problem and thus not guarantee reliable solutions.

In this paper we build on the Uncertain Constraint Satisfaction Problem (UCSP) model, introduced in [20], which aims at providing reliable information by enclosing the data uncertainty with what is known for sure about it, thus guaranteeing a faithful representation of the user's problem. The data enclosure is commonly specified by bounded uncertain coefficients in the problem constraints.

* Present address: SRI International, Menlo Park, CA, USA, nysmith@ai.sri.com

Proceedings of CP'04 Workshop on Modelling and Reformulating Constraint Satisfaction Problems, Toronto, Canada, Sept. 2004. Some minor oversights are corrected in this version.

The outcome derived is such that it infers as much information as can be inferred about the problem, given the present knowledge of the uncertain data. We call it *solution enclosure*, since it can be interpreted as a safe enclosure of the actual but unknown solution; it corresponds in practice to the space of all potential solutions to the UCSP. Given that the data uncertainty can come from erroneous measurements, a central advantage of deriving the solution enclosure is that no optimisation criterion is necessarily required: we do not need to attempt to pick out one 'best' (e.g. most robust) solution. Moreover, in applications such as design, configuration, and diagnosis, it is important to be able to explore the space of solutions systematically whether there is data uncertainty or not (e.g. see [14]) and whether the data is discrete or continuous.

A representative diagnosis LSCO problem, previously modelled as a UCSP, arises in network inference [20]. The traffic volume measurements are uncertain data modelled by closed interval coefficients, known as *parameters*, in the constraints. The uncertain constraints, which model conservation of traffic flow and capacity restrictions, are of linear form, e.g. $[10.0, 15.5]X_i + [5.2, 12.0]X_j + [1.0, 7.5]X_k \leq 100.0$. An efficient means to derive the solution enclosure consists of reformulating the UCSP into a standard linear program (LP), and solving the latter using $2n$ optimisation runs of the Simplex algorithm [19]. This reformulation allowed us to solve a non-linear model (UCSP with parameters multiplied by real variables) in polynomial time. This was possible because the enclosure of the solution to the UCSP at hand describes a convex space that can be mapped to the solution space of a standard LP model. The challenge was thus to reformulate the UCSP into the equivalent LP model, equivalent in terms of the solution space they describe. The UCSP approach allowed us to diagnose the reliability of the previously-used error correction methods.

While we approach these problems from a CP perspective, such issues have been the primary motivation of the field of *reliable computing* (see an overview in [11]). Clearly, such reformulations do not exist for all uncertain constraints. For example, uncertain linear constraints that do not describe a convex solution space can not be reformulated in a manner that preserves the initial solution space (tightness property), and simultaneously be tackled in polynomial time (tractability property) [1].

This paper addresses whether other UCSP models (besides interval LPs) can be reformulated as classical CSPs, in order to guarantee the tightness and tractability properties of respectively the solution enclosure and its derivation. We answer by investigating properties of the constraints as opposed to properties of the solution space described. After providing background (Sections 2 and 3), we present two novel sufficient conditions (Sections 4 and 5), based on the algebraic properties of uncertain constraints, and show how they can be checked in polynomial time for a fixed number of discrete parameters (Section 6). These conditions extend the monotone and row convex properties of classical constraints. With these results, we extend the class of uncertain UCSPs that can be tackled in polynomial time and classify as instances some previously identified constraint classes (Section 7).

2 Background

This section presents the necessary background for the task we address. We introduce the uncertain CSP, which includes parameters to model bounded uncertain data; the

solution enclosure; and the transformation resolution form that reformulates a UCSP into an equivalent CSP, with respect to the solution it describes [20].

2.1 Preliminaries

A classical CSP is a tuple $\langle \mathcal{V}, \mathcal{D}, \mathcal{C} \rangle$, where $\mathcal{V}$ is a finite set of variables, $\mathcal{D}$ is the set of corresponding domains, and $\mathcal{C} = \{c_1, \ldots, c_m\}$ is a finite set of constraints. A solution is a complete consistent value assignment. We describe a CSP by a conjunction of its constraints $\bigwedge_i c_i$ (as opposed to the set of its allowed tuples). Similarly, we represent a solution or set of solutions to a CSP by a conjunction of constraints.

A constraint is a relation between constants, variables and function symbols. The constants we refer to as *coefficients*. A coefficient may be *certain* (its value is known) or *uncertain* (value not known). In a classical CSP, all the coefficients are certain. We call an uncertain coefficient a *parameter*. The set of possible values of a parameter λ_i is its *uncertainty set*, denoted U_i. We say an *uncertain constraint* is one in which some coefficients are uncertain. Observe that the coefficients in an uncertain constraint are still constants; merely as parameters their exact values are unknown. For example, if the parameter λ_1 has uncertainty set $U_1 = \{0,1,2\}$, the constraint $X < \lambda_1$ is uncertain. We assume independence of the parameters.

A *realisation* of the data is a fixing of the parameters to values from their uncertainty sets. We say that any certain constraint corresponding to a realisation is a *realised* constraint, denoted $\hat{c} \in c$. An uncertain constraint can have many realisations, as many as the size of the Cartesian product of the uncertainty sets involved. We write $\mathbb{C}$ for the set of all (uncertain) constraints in a constraint domain, and $\widehat{\mathbb{C}} \subset \mathbb{C}$ for the set of all (certain) realised constraints.

2.2 Uncertain CSP and Solution Enclosures

The *uncertain CSP* extends a classical CSP with an explicit description of the data that allows us to reason with the uncertainty to derive reliable solution enclosures.

Definition 1 (UCSP). *An* uncertain constraint satisfaction problem $\langle \mathcal{V}, \mathcal{D}, \Lambda, \mathcal{U}, \mathcal{C} \rangle$ *is a classical CSP* $\langle \mathcal{V}, \mathcal{D}, \mathcal{C} \rangle$ *in which some of the constraints may be uncertain. The finite set of parameters is denoted by* Λ*, and the set of corresponding uncertainty sets by* $\mathcal{U}$*.*

Example 1. Let X_1 and X_2 both have domains $D_1 = D_2 = [1,3] \subseteq \mathbb{Z}$. Let λ_1 and λ_2 be parameters with uncertainty sets $U_1 = \{0,1,2\}$ and $U_2 = \{1,3,4\}$ respectively. Consider two constraints, both uncertain: $c_1 : X_1 \leq \lambda_1\lambda_2$, and $c_2 : \big(X_1 < X_2 + \lambda_1\big) \wedge \big(\lambda_1 = 2 \Rightarrow X_1 = X_2\big)$. Writing $\mathcal{V} = \{X_1, X_2\}$, $\mathcal{D} = \{D_1, D_2\}$, $\Lambda = \{\lambda_1, \lambda_2\}$, $\mathcal{U} = \{U_1, U_2\}$, and $\mathcal{C} = \{c_1, c_2\}$, then $\langle \mathcal{V}, \mathcal{D}, \Lambda, \mathcal{U}, \mathcal{C} \rangle$ is a UCSP. □

The resolution to a UCSP model P is a *closure*: a set of potential solutions. In this paper we derive the *complete solution enclosure* $\mathrm{Cl}(P)$. This corresponds to all the solutions such that each is supported by *at least one* realisation; each element of $\mathrm{Cl}(P)$ is a *potential solution* and thus should not be excluded. For brevity we call $\mathrm{Cl}(P)$ just the *solution enclosure*. The solution enclosure is not necessarily the outcome to derive when the problem requires an objective function relating to the data (e.g. most

robust solution, covering set closure [8, 20]). For this case, which is beyond the scope of this paper, alternative closures have been defined [18]. Note, however, that when optimisation does not relate to the data uncertainty, deriving the solution enclosure is meaningful, and can be combined with a reformulation approach.

Example 2. Let P be the UCSP of Example 1. The solution enclosure of P in tuple notation is $(X_1, X_2) \in \{(1,1),(1,2),(1,3),(2,2),(2,3),(3,3)\}$. □

By reliable, informally, we mean faithful relative to our knowledge of the state of the real world. The solution enclosure ensures reliability, given present knowledge of the uncertain data, because it is the tightest description of the actual but unknown solution.

2.3 Deriving the Solution Enclosure

One means to derive the solution enclosure of a discrete UCSP (i.e. with discrete uncertainty sets) is to enumerate the realisations. Under each realisation, the UCSP is a classical CSP. By finding all solutions to each such *realised CSP*, and combining them, the solution enclosure of the UCSP is obtained. As observed in [20], naive enumeration can be improved, for example by exploiting algorithms for constructive disjunction [17].

Enumeration of realisations corresponds to enumeration of the values of the parameters. It is worth noting the comparison between enumeration and *hidden variables*. Hidden variables are sometimes used when modelling a LSCO with a classical CSP; these auxiliary variables help state the problem as a CSP but do not appear in the solution [2]. While enumeration of parameter values is operationally similar to labelling of hidden variables, parameters are semantically distinct, because their values cannot be chosen by the user. They are an intrinsic part of the uncertainty captured with a UCSP model, not an auxiliary aid to modelling; indeed, a UCSP may feature hidden variables. The difference in semantics leads to a difference in outcome sought: with parameters, the solution enclosure (all potential solutions), whereas with hidden variables, a complete consistent variable assignment projected onto the non-hidden variables (one solution).

Since the complexity of enumeration can grow rapidly with the size of the uncertainty sets, we seek to reformulate the UCSP into a tractable classical CSP such that the problems are equivalent. An *equivalent CSP*, denoted $\tau(P)$ with respect to a UCSP P, is such that its complete solution set coincides with the solution enclosure of P. A reformulation approach is expected to have much lower complexity than enumeration.

To make precise equivalence of solution sets, we define the partial order $\preceq$ such that: $c_1 \preceq c_2$ iff $\mathrm{Cl}(c_1) \subseteq \mathrm{Cl}(c_2)$. The equivalent CSP should satisfy $P = \tau(P)$ under $\preceq$. Formally, the equivalent CSP is found using a *certain equivalence transform* (CET):

Definition 2 (Certain Equivalence Transform). *A map $\tau : \mathbb{C} \to \widehat{\mathbb{C}}$ is a* certain equivalence transform *if it (1) preserves certainty, i.e. $\tau(\hat{c}) = \hat{c}\ \ \forall \hat{c} \in \widehat{\mathbb{C}}$; and (2) is a closure operator, i.e. is increasing, monotone and idempotent. Increasing means $c \preceq \tau(c)$; if further $\tau(c) \preceq c$, we say τ is a* tight *CET.* □

Thus a CET is tight if the complete solution set of the transformed problem coincides with the solution enclosure of the UCSP; and is non-tight if the former (only)

encloses the latter. Non-tight CETs *correctly* approximate the solution enclosure. They are useful as an approximation when a tight CET is too expensive to compute or the equivalent problem is too expensive to solve exactly.

In this paper we study when we can reformulate a UCSP, either discrete or continuous, into an equivalent, tractable CSP. We seek properties of a constraint class that ensure a tight CET exists and that the equivalent CSP is tractable. For the CET, we want to identify properties that imply Definition 2 and that are easy to check. If the CET or equivalent CSP derived are too costly, then as a secondary aim we seek a non-tight CET.

To achieve both aims, we study the properties of classical constraints to see whether any can be generalised. The intuition is a class of tractable CSP constraints may suggest an analogous class of UCSP constraints where a tight and tractable reformulation exists. We examine the discrete case before generalising to include the continuous case also.

3 Constraint Matrix Representation

Besides the monotone, functional and anti-functional classes of *basic* constraints [16], the tractable classes of classical constraints identified in the literature include *row convex* [15], *max closed* [10], and *connected row convex* (CRC) [6]. Tractable means that there exist polynomial time algorithms for a CSP with constraints of the given class. To identify properties that can be extended for UCSPs, we first extend the constraint matrix representation of a finite domain constraint to constraints with discrete parameters.

3.1 Matrix Representation of a Certain Constraint

Recall that a discrete, binary relation can be represented extensionally with a *constraint matrix*. This representation is useful to characterise properties of a constraint, because algebraic properties of a constraint can be shown equivalent to geometric properties of its constraint matrix representation [6, 15].

Definition 3 (Constraint matrix). *Let R be a binary relation between variables X_i and X_j with domains D_i and D_j respectively. Let M be a $(0,1)$-matrix representing R w.r.t. total orders of D_i and D_j. An element M_{pq} of the matrix is* 0 *iff the tuple of $(D_i)_p$, the p^{th} element of D_i, and $(D_j)_q$, the q^{th} element of D_j, are prohibited by R; otherwise* 1 *denotes a consistent tuple. We say M is a* constraint matrix representation *of R.* □

Example 3. Let X_1 and X_2 both have domains $[1,3] \subseteq \mathbb{Z}$. Let R be the relation $X_1 < X_2$. For the natural ordering of $\mathbb{Z}$, R is represented by the matrix M:

$$\begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \tag{1}$$

Recall now the concept of monotonicity. We write $c(x,y)$ to denote that a certain constraint c is satisfied by the tuple (x,y). A binary constraint is *monotone* iff there exists a total ordering of the domains such that, for all x,y, $c(x,y) \Rightarrow c(x',y') \; \forall x' < x$ and $y' > y$ [16]; here $\Rightarrow$ denotes logical implication as usual. [15] show a constraint is monotone iff $M_{pq} = 1 \Rightarrow M_{p'q} = 1 \; \forall p' \leq p$ and $M_{pq} = 1 \Rightarrow M_{pq'} = 1 \; \forall q' \geq q$. Observe the constraint $X_1 < X_2$ is monotone and its constraint matrix (1) obeys this property. □

3.2 Matrix Representation of an Uncertain Constraint

We now define the matrix representation of an uncertain constraint. We use it to help prove sufficient conditions for a CET in the discrete case. We can, optionally, also use it to identify constraints which satisfy these conditions.

Definition 4 (Parameter constraint matrix). *Let $c \in \mathbb{C}$ be an uncertain constraint with two discrete variables and one discrete parameter. For a given ordering of the domains D_i and D_j of the variables and the uncertainty set U of the parameter λ, a* constraint matrix *for c is a $(0,1)$-matrix M on three dimensions, where $M_{pqr} = 1 \iff X_i = (D_i)_p \wedge X_j = (D_j)_q$ is a consistent tuple for c under the realisation $\lambda = (U)_r$.* □

Thus a matrix representation of an uncertain constraint has one classical constraint matrix for each realised constraint. It is important to note a constraint matrix depends on the orderings for $\mathcal{D}$ and Λ. Thus, as in the classical case, there may be many matrices representing a constraint. We say a 'column' M_{pqr} as r varies is a *file* of M.

Example 4. Let X_1 and X_2 both have domains $[1,3] \subseteq \mathbb{Z}$, and let λ_1 be a parameter with uncertainty set $\{0,1,2\}$. Let c be the uncertain constraint $X_1 < X_2 + \lambda_1$. For the natural ordering of $\mathbb{Z}$, c is represented by the matrix M:

$$\left(\begin{vmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{vmatrix} \begin{vmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{vmatrix}\right) \tag{2}$$

where we have written each *sheet* of the third dimension of the matrix in order from left to right. In the first sheet $\lambda_1 = 0$, in the second $\lambda_1 = 1$, and in the last, $\lambda_1 = 2$. □

4 Two Sufficient Conditions

Recall the properties we seek must imply Definition 2, to give a sufficient condition that a CET exists for the corresponding constraint classes. Moreover, we want to identify properties that are useful in practice: leading to a simple reformulation (i.e. both a simple transformation and a tractable equivalent problem), easy to check, and holding for a broad range of classes. We begin with binary constraints and a single parameter.

To specify a CET, we must specify $\tau(c)\ \forall c$. Consider a constraint matrix M_c of an uncertain constraint $c \in \mathbb{C}$. Observe that a tuple for the variables is a potential solution iff there exists a 1 in the corresponding file of M_c. Thus a CET can be specified by: $(M_{\tau(c)})_{pq} = 1$ if $\exists r$ s.t. $(M_c)_{pqr} = 1$. If further the converse holds, i.e. the last condition is 'iff', then the CET is tight. Without additional knowledge, however, to compute this CET we must search for r s.t. $(M_c)_{pqr} = 1$, which is essentially enumerating the realisations. By restricting the constraint class, we seek properties that produce a more practical CET. We want a simple means of knowing whether there is a 1 in each file.

We show that a distinguishing feature of monotonicity and row convexity, when translated into parameters, is that they ensure contiguous bands of 1s in the files of the constraint matrix. For *parameter monotone*, we will show we can obtain a tight CET by looking at the last sheet. This means we can derive the CET without enumeration, and moreover without needing a matrix representation of the constraint. In contrast, when

we consider other properties of certain constraints, such as functional, max closed, and CRC, we observe that non-contiguous 1s may arise in a file. This means we cannot easily say where to look, to see whether there is a 1 in a file or not. We must search through the elements of a file, which returns us to enumeration of realisations. Thus there is no easy way to detect and derive a CET.

4.1 Parameter Monotone

We now give a first property, the analogue of monotone, that produces a CET that is simple and easy to check. For $c \in \mathbb{C}$, we write $\hat{c}(x,y;\lambda)$ to denote that c under the realisation $\lambda_1 = \lambda$, written $\hat{c}_\lambda$, is satisfied by the tuple (x,y).

Definition 5 (Parameter monotone). *Let $c \in \mathbb{C}$ be an uncertain constraint with two discrete variables and one discrete parameter. We say that c is* parameter monotone *if there exists a total ordering of the uncertainty set of the parameter such that:*

$$\forall \lambda \in U_1,\ \ \hat{c}(x,y;\lambda) \Rightarrow \hat{c}(x,y;\lambda')\ \forall \lambda' > \lambda \tag{3}$$

For example, the constraint $X_1 < X_2 + \lambda_1$ is parameter monotone, but the constraint $X_1 + X_2 \neq \lambda_1$ is not. Note that whether the realisations of an uncertain constraint are monotone is unrelated to whether the constraint is parameter monotone: the former concerns the variables (one sheet of the constraint matrix), while the latter concerns the parameters (one file through all sheets).

The idea of the CET is: use the last sheet of a constraint matrix of each constraint. For any uncertain constraint, this is a single, certain constraint that is easy to derive. Moreover, given the ordering of the uncertainty set, this CET can be derived without knowing the constraint matrix: we simply take the realised constraint corresponding to the greatest λ value, as we will explain. In order to prove this CET for parameter monotone constraints exists and is correct and tight, we link the algebraic definition of parameter monotone with a geometric property of constraint matrices:

Proposition 6 (Parameter monotone constraint matrix). *Let $c \in \mathbb{C}$ have a constraint matrix M. There exists an ordering of the domains and uncertainty sets of c such that:*

$$\forall p,q,\ \ M_{pqr} = 1 \Rightarrow M_{pqr'} = 1\ \forall r' \geq r \tag{4}$$

iff c is parameter monotone.

Proof. Suppose first that the property (4) holds for a constraint matrix of c. Consider any file M_{pq}. The elements in this file are a vector of form $(0,\ldots,0,1,\ldots,1)$, i.e. a sequence of 0s followed by a sequence of 1s (either but not both sequences may be empty). Thus for any two realisations $\lambda_1 < \lambda_2$ of λ, we have $\hat{c}_{\lambda_1} \Rightarrow \hat{c}_{\lambda_2}$ (since $0 \Rightarrow 1$ and $1 \Rightarrow 1$). But this is exactly the definition of parameter monotone in Definition 5. The converse is argued similarly. □

For example, the constraint $X_1 < X_2 + \lambda_1$ from Example 4 is parameter monotone and its constraint matrix (2) obeys the above property.

Proposition 6 is the basis of the proof that parameter monotonicity does lead to the CET claimed above. Observe first the link between implication of uncertain constraints and their ordering by $\preceq$. Logical implication of uncertain constraints is defined by [20]: if every assignment that satisfies some realisation of c_1 also satisfies some (not necessarily the same) realisation of c_2, then c_1 implies c_2. Observe that for $c_1, c_2 \in \mathbb{C}$, if $c_1 \Rightarrow c_2$ then $c_1 \preceq c_2$. This means $c \Rightarrow \tau(c)$ is a sufficient condition that $c \preceq \tau(c)$, which is a key step in the proof that the CET is correct.

Before giving the proof formally, we need a lemma describing when multiple constraints are parameter monotone. It may be that two constraints are both parameter monotone, but only under incompatible orderings. Therefore we say two constraints are *simultaneous* parameter monotone if there exists an ordering of the uncertainty sets involved such that both constraints are parameter monotone.

Lemma 7 (Simultaneous parameter monotone closed under conjunction). *A conjunction of simultaneous parameter monotone constraints is parameter monotone.*

Proof. By induction, it suffices to look at the conjunction of two constraints, c_1 and c_2. By hypothesis, both are parameter monotone under the same ordering. This means there exists an ordering on $\mathcal{U}$ such that, for any values of the variables in the scope of the constraints, the files in the constraint matrices M_{c_1} and M_{c_2} are vectors of the form $(0,\ldots,0,1,\ldots,1)$. If we take the conjunction of two such vectors, the result is a vector of the same form, since a 1 can occur iff there is a 1 in both vectors, but then all subsequent elements in both must be 1. This shows that the constraint matrix of $c_1 \wedge c_2$ obeys (4), and so by Proposition 6, $c_1 \wedge c_2$ is parameter monotone. □

For uncertainty sets drawn from domains such as $\mathbb{Z}$, where there is a natural total order, simultaneous parameter monotonicity will frequently hold. Indeed, for such sets, all constraints that are parameter monotone with respect to the natural order are simultaneous parameter monotone. We are now able to prove the main result:

Proposition 8 (CET for parameter monotone constraints). *If $\mathcal{C}$ consists of simultaneous parameter monotone constraints, then there exists a tight CET for P.*

Proof. Suppose $c \in \mathcal{C}$ is parameter monotone. There exists an ordering of the uncertainty set of λ such that the constraint matrix M_c of c satisfies (4). Let $\overline{\lambda}$ be the greatest value of λ under this ordering, and define $\tau(c)$ to be c under the realisation $\overline{\lambda}$, which we denote $\hat{c}_{\overline{\lambda}}$. By Definition 5 and Proposition 6, we have $\hat{c} \Rightarrow \tau(c)\, \forall \hat{c} \in c$, and so $c \preceq \tau(c)$. Thus the map τ_c defined for c is increasing, and it is straight-forward to verify the other properties of Definition 2 to confirm it is a CET. Moreover, $\tau(c)$ is a tight CET, since $\tau(c) \Rightarrow \hat{c}$ for some $\hat{c} \in c$, which means that $\tau(c) \preceq c$.

Now suppose we thus define τ_c for each $c \in \mathcal{C}$. We must show we can extend the τ_c defined for the individual constraints to a CET for the combined constraint $\mathcal{C} = \bigwedge_i c_i$. Since $\mathcal{C}$ consists of simultaneous parameter monotone constraints by hypothesis, $\mathcal{C}$ is parameter monotone by Lemma 7. Thus there exists a τ for $\mathcal{C}$ and, by the proof of Lemma 7, it is $\tau(\mathcal{C}) = \bigwedge_i \tau_{c_i}(c_i)$. Further, for the same reason as above, τ is tight. □

Example 5 (Example 4 continued). The parameter monotone constraint $X_1 < X_2 + \lambda_1$ has CET $X_1 < X_2 + \overline{\lambda_1}$, i.e. $X_1 < X_2 + 2$. Observe that the sheet of the constraint matrix (2) corresponding to $\lambda_1 = 2$ is the constraint matrix of $\tau(c)$. □

In Section 5 we extend parameter monotonicity to many variables and parameters. However even for binary constraints and one parameter, it is a useful concept in practice:

Example 6. In a Simple Temporal Problem with Uncertainty (STPU) [13], the constraints have the form $|Y - X| \leq \lambda$, where λ has uncertainty set given by an interval. A system of STPU constraints can be viewed as a UCSP. Its minimal network, which can be obtained in polynomial time, is equivalent to the complete solution enclosure. The tractable derivation of the solution enclosure is explained because STPU constraints are parameter monotone. □

Example 7. Consider a set $\mathcal{C}$ of binary monotone constraints. When $\mathcal{C}$ is simultaneous parameter monotone, a CET is obtained simply by taking the relevant bounds of uncertainty sets, by Proposition 8. Each $c \in \mathcal{C}$ is transformed into a binary monotone constraint. The complete solution set of the latter can be found in linear time (in the number of variables) with a 2D integer hull algorithm [7]. For example, $\{-3,0,3\}X \leq 2Z + \{2,3,5\}$ is transformed to $-3X \leq 2Z + 5$, for $X \geq 0$. Even for binary constraints, note the parameter monotone property is required for this CET. A monotone basic constraint in general need not be parameter monotone; consider the constraint given by:

$$\left(\begin{vmatrix} 1 & 1 \\ 0 & 0 \end{vmatrix} \begin{vmatrix} 0 & 1 \\ 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 1 \\ 1 & 1 \end{vmatrix}\right) \tag{5}$$

4.2 Parameter Row Convex

We now identify a second property for the existence of a CET. This property, the analogue of row convex, is a generalisation of parameter monotone. It applies to more classes of constraints but produces a more complicated CET. Recall from [15] that a relation is *row convex* if in every row of its constraint matrix, all the 1s are consecutive.

Definition 9 (Parameter row convex). *Let $c \in \mathbb{C}$ be an uncertain constraint with two discrete variables and one discrete parameter. We say that c is* parameter row convex *if there exists a total ordering of the uncertainty set of the parameter such that in every file of a constraint matrix of c, all the* 1*s are consecutive. Geometrically:*

$$\forall p, q,\ \exists r_1 \leq r_2 \text{ s.t. } r_1 \leq r \leq r_2 \iff M_{pqr} = 1 \tag{6}$$

Parameter row convexity generalises parameter monotonicity, just as row convexity generalises monotonicity. Intuitively, an uncertain constraint c is parameter row convex if there is a contiguous band of 1s in every file of its constraint matrix; if, in every file, this band extends to the greatest λ value, then c is parameter monotone.

Example 8. Let X_1 and X_2 both have domains $[1,3] \subseteq \mathbb{Z}$, and let λ_1 be a parameter with uncertainty set $\{0,1,2\}$. Let c be the uncertain constraint $\big(X_1 < X_2 + \lambda_1\big) \wedge \big(\lambda_1 = 2 \Rightarrow X_1 = X_2\big)$. For the natural ordering of $\mathbb{Z}$, c is represented by the matrix M:

$$\left(\begin{vmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{vmatrix} \begin{vmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}\right) \tag{7}$$

where we have written the matrix as before. c is not parameter monotone, because $M_{132} = 1$ but $M_{133} = 0$, but it is parameter row convex. □

The CET for parameter row convex constraints extends the idea used for parameter monotone constraints. For the latter we used $\hat{c}$ under the realisation $\overline{\lambda}$; the CET was independent of the value of the variables. Since now we may have $\hat{c}_\lambda \not\Rightarrow \hat{c}_{\overline{\lambda}}$ for some realisation λ (if there are p,q such that $M_{pq\lambda} = 1$ but $M_{pq\overline{\lambda}} = 0$), we cannot simply use the last sheet $\overline{\lambda}$ of the constraint matrix. Instead, we define the CET depending on the value of the variables. For each tuple, we use the maximal λ_{pq} such that $M_{pq\lambda_{pq}} = 1$; such a λ_{pq} uniquely exists for each file by the parameter row convex property.

We say two constraints are *simultaneous* parameter row convex if there exists an ordering of the uncertainty sets involved such that both constraints are parameter row convex, and for each (p,q), $(\lambda_1)_{pq} = (\lambda_2)_{pq}$. The second part of the condition means the maximal λ_{pq} is the same for both constraints, for any tuple of the variables.[1] This means, according to the proof of the next theorem (omitted for lack of space, but along the lines of Proposition 8), that we can define $\tau(c)$ to be the conjunction:

$$\bigwedge_{p,q} \left(X_1 = p \wedge X_2 = q \Rightarrow \hat{c}_{\lambda_{pq}\downarrow pq}\right) \tag{8}$$

where $c\downarrow_{pq}$ is the projection of a certain constraint $c \in \widehat{\mathbb{C}}$ onto the tuple $X_1 = p \wedge X_2 = q$.

Proposition 10 (CET for parameter row convex constraints). *If $\mathcal{C}$ consists of simultaneous parameter row convex constraints, then there exists a tight CET for P.* □

Example 9 (Example 8 continued). The parameter row convex constraint $\left(X_1 < X_2 + \lambda_1\right) \wedge \left(\lambda_1 = 2 \Rightarrow X_1 = X_2\right)$ is transformed by the CET (8) into the conjunction of:

$$\begin{array}{rcl}
X_1 = 1 \wedge X_2 = 1 & \Longrightarrow & X_1 < X_2 + 2 \\
X_1 = 2 \wedge X_2 = 1 & \Longrightarrow & \bot \\
X_1 = 3 \wedge X_2 = 1 & \Longrightarrow & \bot \\
X_1 = 1 \wedge X_2 = 2 & \Longrightarrow & X_1 < X_2 + 1 \\
X_1 = 2 \wedge X_2 = 2 & \Longrightarrow & X_1 < X_2 + 2 \\
X_1 = 3 \wedge X_2 = 2 & \Longrightarrow & \bot \\
X_1 = 1 \wedge X_2 = 3 & \Longrightarrow & X_1 < X_2 + 1 \\
X_1 = 2 \wedge X_2 = 3 & \Longrightarrow & X_1 < X_2 + 1 \\
X_1 = 3 \wedge X_2 = 3 & \Longrightarrow & X_1 < X_2 + 2
\end{array} \tag{9}$$

which reduces to the more compact:

$$\left(X_1 = X_2 \Rightarrow X_1 < X_2 + 2\right) \wedge \left(X_1 \neq X_2 \Rightarrow X_1 < X_2 + 1\right) \tag{10}$$

While parameter row convexity is defined in terms of a constraint matrix representation, the CET (8) is independent of it. However, to produce the CET we need to know

[1] In contrast to classical row convexity, for parameter row convex constraints it is not true that a local intersection point implies a global intersection point. That is, w.r.t. a given ordering, any pair of parameter row convex vectors will have a 1 in their conjunction, but this 1 need not be in the same place between different pairs. Hence the second part of the condition is required.

the λ_{pq} and, without additional knowledge about the constraints, a matrix representation may be necessary to find the λ_{pq}. This is in contrast to parameter monotone, where to produce the CET we need only know the ordering of the uncertainty set.

Moreover, the CET for a general parameter row convex constraint is a conjunction, with as many terms as the size of the Cartesian product of the variable domains. This is too unwieldy to be useful unless we can combine some of the conjuncts, as in Example 9. When we can combine *all* conjuncts, so the head of the clause in (8) is the universal constraint, then we have a parameter monotone constraint.

5 Multiple Parameters

The above results were stated for discrete binary constraints with a single parameter. It is not difficult to see they hold also for unary constraints; the question is how to generalise to non-binary constraints and to multiple parameters.

We generalise to non-binary uncertain constraints readily, because neither the definition of parameter monotone (Definition 5) nor of parameter row convex (Definition 9) depends on the number of variables. Each sheet of a constraint matrix is now itself an n-dimensional matrix, where n is the number of variables. Although this hampers one's intuition, the algebraic results are unaffected.

We generalise to multiple parameters as follows. [15] generalise a property to classical non-binary constraints by stating: a property holds for a non-binary constraint $c \in \widehat{\mathbb{C}}$ if, for every pair of variables, the corresponding property holds for the binary constraint obtained by projecting onto that pair, i.e. if $c{\downarrow_{pq}}$ obeys the property $\forall p,q$.

To make a similar generalisation we must define the projection, or *restriction*, of an uncertain constraint to one parameter. For $c \in \mathbb{C}$, let $c{\downarrow_\lambda}$ be a restriction of c to the parameter λ: a one-parameter constraint obtained by considering all other parameters to be constants, for some tuple of values from their uncertainty sets. Note only the mutually consistent values for the other parameters (those that obey all data constraints) need be considered. Then we can say a property holds for an uncertain constraint c with $\ell \geq 1$ parameters if, for every parameter λ, the corresponding simultaneous property holds for *all* restrictions $c{\downarrow_\lambda}$. If u is the maximum size of an uncertainty set, then the number of restricted constraints of c is at most $O(\ell u^{\ell-1})$.

Informally, a property holds for an ℓ-parameter uncertain constraint if, in its multi-dimensional constraint matrix, each slice parallel to one parameter axis and to all of the variable axes obeys the property, and the orderings of the uncertainty sets are compatible. This is a strong extension in that it requires the constraint to be simultaneously parameter monotone (resp. parameter row convex) for all restrictions to one parameter.

Example 10. Consider the constraint c: $X \leq \lambda_1\lambda_2$, where λ_1 and λ_2 have uncertainty sets $\{0,1,2\}$ and $\{1,3,4\}$ respectively. A restriction of c to λ_1 is $X \leq 3\lambda_1$; there are three such restrictions $c{\downarrow_{\lambda_1}}$, one for each value of λ_2 in U_2. Now if either parameter is assumed to be a constant (any value from its uncertainty set) then the resulting restricted constraint is parameter monotone; every restriction of c is thus parameter monotone. Moreover, since all the restrictions $c{\downarrow_{\lambda_1}}$ use the same total order of U_1, they are simultaneous parameter monotone; likewise for the $c{\downarrow_{\lambda_2}}$. Hence c is parameter monotone.

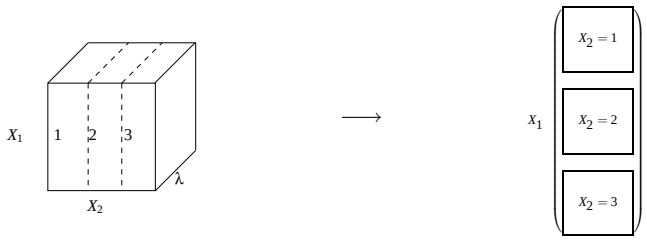

Figure 1. Slicing the matrix of a binary uncertain constraint. The left-hand cube is the parameter constraint matrix; the files are depicted on the z-axis. The matrix is sliced vertically along the values of X_2, and rearranged to form the right-hand matrix

The consequence of this strong extension is that the CET sufficiency results extend simply, provided the parameters are independent. For a parameter monotone constraint c, we have $\tau(c) = \hat{c}$ where $\hat{c}$ corresponds to c under the realisation $(\overline{\lambda_1}, \ldots, \overline{\lambda_\ell})$, i.e. the greatest value of each of the parameters independently.

For a parameter row convex constraint, let v be a tuple of values for the variables. In the binary case, $v = (p,q)$ as before. We have $\tau(c) = \hat{c}$ where $\hat{c}$ corresponds to the realisation $((\lambda_1)_v, \ldots, (\lambda_\ell)_v)$. Here, $(\lambda_i)_v$ is the greatest value of λ_i such that $M_{v\lambda_i} = 1$.

In general, however, multiple parameters will not be independent. Lack of space forces us to sketch the consequences. For parameter monotonicity (but not parameter row convexity), it can be shown that the above CET is still correct, but is no longer tight. This means that a non-tight CET is found by assuming the parameters are independent. For instance, consider Example 10 with the additional constraint $\lambda_1 + \lambda_2 \leq 4$.

6 Checking the Sufficient Conditions

We have stated and proved two sufficient conditions for the existence of a CET: parameter monotone and parameter row convex. We have shown how they apply to discrete uncertain constraints with arbitrary numbers of variables and parameters. We now give a method to test whether these properties hold for a binary uncertain constraint.[2]

One way to determine whether the parameter monotone property holds for a constraint is to use its characterisation in terms of the constraint matrix, Proposition 6. To check this condition for $c \in \mathbb{C}$, we need to find a simultaneous ordering of the uncertainty sets such that each file of the constraint matrix M_c is a sequence of zeros followed by a sequence of ones. An elegant result from computational graph theory states that for a $m \times n$ $(0,1)$-matrix with k non-zero entries, we can test in linear time ($O(m+n+k)$) for the existence of a permutation of the columns such that the matrix is row convex [4]. Granted the constraint matrix, we can use this result to test for both parameter monotonicity and row convexity, for each parameter in turn.

Suppose first there is one parameter. We take the matrix M_c and cut it into parallel slices along the parameter dimension, as Fig. 1 illustrates. This gives us $|D_2|$ slices of size $|D_1| \times |U_1|$ each. Arrange the slices in a column, to give a $|D_1| \times |D_2| \times |U_1|$

[2] An open issue is whether the method can be extended for n-ary uncertain constraints.

matrix, and test for row convexity. By permuting the columns of the assembled matrix, we test for a permutation of the uncertainty set of the parameter. If a permutation is found that makes the assembled matrix row convex, this corresponds to an ordering on the uncertainty set such that each file of the original constraint matrix M_c is row convex. This means c is parameter row convex. Of course, c may be parameter monotone but not parameter row convex. We can test directly for monotonicity by adapting the algorithm of [4] to require the consecutive ones to finish at the end of each row, and not before.

Suppose now there are multiple parameters. We perform the above procedure for each restricted constraint. If all pass, we cannot yet conclude c is parameter row convex, because the restricted constraints must be simultaneous row convex. Thus we must additionally test that the same permutation is used for each restricted constraint. This suffices to prove parameter monotonicity. For parameter row convexity, the definition of simultaneous further requires that the λ_{pq} agree for each file. We can check this at cost $O(d^2)$, where d is the maximum domain size, for each pair of restricted constraints. Both these additional tests may be performed incrementally between each restricted constraint and the last. In total, the test takes linear time in the size of the product of the variable domains and the uncertainty set ($O(d^2 + u + ud^2) = O(ud^2)$) for a one-parameter constraint. Thus with ℓ parameters, the total time complexity is $O(\ell u^\ell d^2)$.

7 Reformulation for Continuous UCSPs

Parameter monotonicity and row convexity can both produce a tight CET. For both, the CET is defined independently of any constraint matrix representation. The matrix representation is a tool useful in identifying the conditions in the discrete case. Because it is not necessary for their definition, in principle both conditions apply to continuous constraints, i.e. to a UCSP with continuous uncertainty sets.

For parameter monotone, the CET simply takes the constraints under their maximal realisations. This is done for continuous parameters as easily for discrete parameters. For parameter row convex, the CET depends on the value of the variables and thus may be more complicated: both derivation of the CET itself and the form of the equivalent problem it yields. We have seen that this CET may simplify, but also that we may need to know a matrix representation to derive it.

Tractability of Polynomial Inequality Constraints Motivated by a real-world UCSP, we now exhibit a class of constraints where we can prove a priori that parameter row convexity holds. The class we consider is polynomial inequality constraints, such as $\lambda_1 X + \lambda_2 Y \leq \mu$, where λ_i and μ are parameters. This is an instance where the CET does indeed simplify enough to become practical.

Proposition 11 (Parameter row convex CET for polynomial inequality constraints). *Let $\mathcal{C}$ consist of polynomial inequality constraints over $\mathbb{Z}$, where the coefficients of each term are single parameters and each constraint features at least one variable. If the uncertainty sets can be simultaneously totally ordered, then there exists a tight CET.* □

For lack of space the proof is omitted. It constructs explicitly the CET. Taking a general $c \in \mathcal{C}$, for each tuple of values for the variables, we can partition the parameters

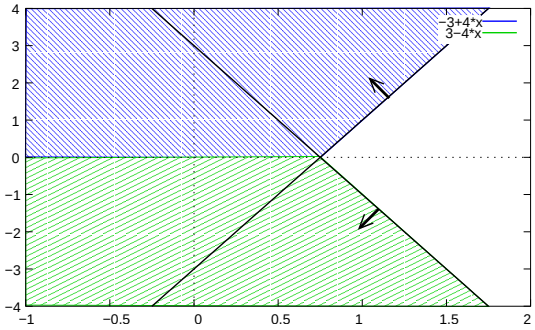

Figure 2. Polynomial inequality CET described by two constraints, shown by the upper and lower shaded areas; the solution to every realised constraint lies in one of them

into two sets. The 'left-hand' parameters λ we minimise to their lower bounds $\underline{\lambda}$; the 'right-hand' parameters μ we maximise to their upper bounds $\overline{\mu}$.

Example 11. Consider the UCSP with variables $X, Y \in \mathbb{Z}$ and the single constraint:

$$\lambda_1 X + \lambda_2 Y \leq \mu \tag{11}$$

where the uncertainty sets for the parameters are integer intervals: $U_{\lambda_1} = [4,5]$, $U_{\lambda_2} = [-1,0,1]$, $U_\mu = [2,3]$. The CET of Proposition 11 transforms (11) into a classical constraint according to:

$$\begin{cases} Y \geq -3 + 4X & \text{if } Y \geq 0 \\ Y < 3 - 4X & \text{if } Y < 0 \end{cases} \tag{12}$$

Thus we transform a parameter row convex uncertain constraint into a piecewise monotone constraint, which is depicted in Fig. 2. Across all the assignments for the variables, only two partitions of the parameters arise for (11). These are the two cases of (12). In both cases, the transformed constraint $\tau(c)$ involves only extremal values of the parameters. In fact, λ_1 is always a 'left-hand' parameter and μ always a 'right-hand' parameter. λ_2 is on the left- or right-hand side depending on the sign of Y. □

It can be shown that Proposition 11 holds also for similar constraints over $\mathbb{R}$ [18]. In particular, linear constraints over the reals with non-negative variables are a special case of Proposition 11. The domain constraints $X \geq 0$ restrict the solution space to the positive orthant. The CET is given by a single case, which depends only on the sign of the coefficients [19]. We can see this situation in Fig. 2 restricted to the positive orthant: when $X, Y \geq 0$, the solution enclosure of the UCSP is given by $Y \geq -3 + 4X$. The network inference LSCO problem introduced earlier features constraints of this class.

The CET (8) for parameter row convex constraints involves one term for each tuple of variable values. As we remarked earlier, in general it is not practical to form a CET by considering each such element of $\mathcal{D}$. Often, however, the CET turns out to be described by a small number of cases given by ranges of values, as Example 11 illustrates. The next example shows another instance of Proposition 11 where the CET involves many fewer terms than the general case.

Example 12. Let X_j be variables over $\mathbb{Z}$ or $\mathbb{R}$. Consider a polynomial arithmetic constraint with a single parameter as follows: $\sum_i a_i \prod_j X_j^{e_{ij}} \leq \mu$, where $a_i, e_{ij} \in \mathbb{N}$ are constant coefficients. Proposition 11 tells us this constraint is parameter row convex. Moreover, in fact it is parameter monotone, with CET $\sum_i a_i \prod_j X_j^{e_{ij}} \leq \overline{\mu}$, where $\overline{\mu}$ is the greatest value in U_μ under the natural order. □

8 Conclusion and Future Work

In this paper we have addressed the challenge of tractably solving an uncertain CSP for its solution enclosure, by reformulating the problem into an equivalent CSP. We have identified two sufficient conditions on the constraint class, parameter monotone and parameter row convex, which ensure a reformulation into a tractable CSP. The transformation is achieved by a CET, whose existence and practicality is guaranteed by these properties. It transforms the UCSP into an equivalent classical CSP, whose complete solution set is the solution enclosure sought. Thus solving the UCSP is tractable provided deriving this latter solution set is.

Parameter monotonicity is useful because it tells us a CET exists when we take the extremal values of the uncertainty sets, usually under the natural order or its reverse. Parameter row convexity is more general, but its use depends on how many of the conjuncts in its CET combine; it is practical if only a few terms remain, such as in Example 11. It is important to note, however, that both conditions we proved are sufficient but not necessary. For example, while constraint (5) is not parameter row convex, a CET is given by its realisation under $\overline{\lambda}$.

The two conditions suggest situations when reformulation of an uncertain CSP is more effective than direct solving based on enumerating the realisations. The solution enclosure can be derived without considering all realisations, and hence the complexity is much less. Although the algorithm we gave to check the conditions works with a constraint matrix representation (which is a form of enumeration), the CETs described in this paper do not depend on any matrix representation. Nonetheless, unless we have specific knowledge about the constraints (e.g. in Example 11, on the sign of the variables), to derive the CET for a parameter row convex constraint may require some search through its matrix representation.

Since a UCSP is a restricted form of a quantified CSP (QCSP), our contribution can be seen as an efficient solving method for a subclass of QCSPs with practical applications. We would like to investigate whether generic QCSP solvers (e.g. [12]) can be adapted to exploit the restricted quantification found in a UCSP. In future work we also plan to examine other properties of the constraints in a UCSP that make it conducive to reformulation. For example, we wish to explore the structure of the constraint graph, to see whether analogous results be drawn for UCSPs as for CSPs, for instance on bounded tree width.

Acknowledgements. The authors thank W. Harvey, A. Sadler, and T. Winterer for discussions, and the reviewers for their suggestions. This work was partially supported by the EPSRC under grant GR/N64373/01.

References

[1] G. Alefeld, V. Kreinovich, and G. Mayer. On the solution sets of particular classes of linear systems. *J. Computational and Applied Mathematics*, 152:1–15, 2003.

[2] F. Bacchus, X. Chen, P. van Beek, and T. Walsh. Binary vs. non-binary constraints. *Artificial Intelligence*, 140(1–2):1–37, 2002.

[3] Y. Ben-Haim. Set-models of information-gap uncertainty: Axioms and an inference scheme. *J. Franklin Institute*, 336:1093–1117, 1999.

[4] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using P-Q tree algorithms. *J. Computational and Systems Science*, 13:335–379, 1976.

[5] M. Christie, E. Languénou, and L. Granvilliers. Modeling camera control with constrained hypertubes. In *Proc. of CP'02*, LNCS 2470, pages 618–632, Ithaca, NY, Sept. 2002.

[6] Y. Deville, O. Barette, and P. Van Hentenryck. Constraint satisfaction over connected row convex constraints. *Artificial Intelligence*, 109(1-2):243–271, 1999.

[7] W. Harvey. Computing two-dimensional integer hulls. *SIAM J. Computing*, 28(6):2285–2299, Aug. 1999.

[8] E. Hebrard, B. Hnich, and T. Walsh. Super solutions in constraint programming. In *Proc. of CP-AI-OR'04*, pages 157–172, Nice, France, Apr. 2004.

[9] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis*. Springer, 2001.

[10] P. Jeavons and M. Cooper. Tractable constraints on ordered domains. *Artificial Intelligence*, 79(2):327–339, 1995.

[11] R. B. Kearfott. Interval computations: Introduction, uses, and resources. *Euromath Bulletin*, 2(1), 1996.

[12] N. Mamoulis and K. Stergiou. Algorithms for quantified constraint satisfaction problems. In *Proc. of CP'04*, Toronto, Canada, Sept. 2004.

[13] P. Morris, N. Muscettola, and T. Vidal. Dynamic control of plans with temporal uncertainty. In *Proc. of IJCAI'01*, pages 494–502, Seattle, WA, Aug. 2001.

[14] D. Sam-Haroud and B. Faltings. Consistency techniques for continuous constraints. *Constraints*, 1(1/2):85–118, 1996.

[15] P. van Beek and R. Dechter. On the minimality and global consistency of row-convex constraint networks. *J. ACM*, 42:543–561, 1995.

[16] P. Van Hentenryck, Y. Deville, and C.-M. Teng. A generic arc-consistency algorithm and its specializations. *Artificial Intelligence*, 57(2–3):291–321, 1992.

[17] P. Van Hentenryck, V. A. Saraswat, and Y. Deville. Design, implementation, and evaluation of the constraint language cc(FD). *J. Logic Programming*, 37(1–3):139–164, 1998.

[18] N. Yorke-Smith. *Reliable Constraint Reasoning with Uncertain Data*. PhD thesis, IC-Parc, Imperial College London, June 2004.

[19] N. Yorke-Smith and C. Gervet. Data uncertainty in constraint programming: A non-probabilistic approach. In *Proc. of AAAI 2001 Fall Symposium on Using Uncertainty within Computation*, Nov. 2001. Available at: `www-users.cs.york.ac.uk/~tw/fall/`.

[20] N. Yorke-Smith and C. Gervet. Certainty closure: A framework for reliable constraint reasoning with uncertainty. In *Proc. of CP'03*, LNCS 2833, pages 769–783, Sept. 2003.