Certainty Closure

Reasoning About Constraint Problems with Uncertain Data

Neil Yorke-Smith and Carmen Gervet

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

Abstract. We present a generic framework to reason about constraint problems with incomplete or erroneous data. Such problems are often simplified at present to tractable deterministic models, or modified using error correction methods, with the aim of seeking a solution. However, this can lead us to solve the wrong problem because of the approximations made. Such an outcome is of little help to the user who expects the right problem to be tackled and correct information returned. The certainty closure framework aims to provide the user with reliable insight by: (1) enclosing the uncertainty using what is known for sure about the data, (2) guaranteeing the true problem is contained in the model so described, and (3) efficiently deriving the closure to an uncertain constraint problem. In this paper we define the certainty closure and show how it can be derived by transformation to an equivalent certain problem. We demonstrate the benefits of the framework on a real-world network traffic analysis problem with uncertainty.

1 Motivation

Real-world Large Scale Combinatorial Optimisation problems (LSCOs) have inherent data uncertainties. The uncertainty can be due to the dynamic and unpredictable nature of the commercial world (e.g. [4]), but also due to the information available to those modelling the problem. In this paper we are concerned with the latter form of uncertainty, which can arise when the data is not fully known or is even erroneous. Our work is motivated by practical issues we faced when working on two real-world applications: energy trading [12] and network traffic analysis [13].

In both applications the data information is incomplete or erroneous. In the energy trading problem, the demand and cost profiles have evolved due to market privatisation; thus the existing simulation or stochastic models would not help address the actual problem, since no valid data trends are available. The data is obsolete and inconsistent with the constraint model. In the network traffic analysis problem, we are forced to use partial data due to the overwhelming amount of information. Further, the data can be erroneous due to unrecorded packet loss or time gaps between readings of router tables.

When addressing the energy trading problem, we understood that the customer did not need nor want a solution to an approximation of his problem, but rather help to formulate the right problem and then solve it. This might require dealing with approximation models, but the ultimate goal becomes to refine the problem definition as opposed to solely providing a solution. It became clear that research work was necessary to extend the potential of constraint programming to deal with problem refinement.

In the face of data uncertainty, a set of CSP derivatives has been defined to model the uncertainties by approximating the state of the real world and building a solution to solve the problem (and possibly deriving a rational decision making procedure, such as expected utility). The technological support is currently a means to solve an LSCO problem by various approximations: for example, error correction methods to ensure feasibility. The goal, as far as we know, has not been explicitly to learn about the problem and refine its uncertain definition. Some of the current approaches could be used for such a purpose by placing the burden on the user: the user runs simulations and attempts to refine his problem statement by analysing the relationships between problems and solutions produced. However the practicality of this approach (in terms of convenience and of polynomial algorithms) remains to be shown.

With the *certainty closure* framework we focus on uncertainty in data where we know that the true problem is satisfiable. Due to data collection methods leading to errors or constraining us to deal with incomplete data, however, the CSP model of the problem is unsatisfiable. We propose to, first, ensure that the true problem is embedded in the model, i.e. remove approximations or assumptions about the data and enclose the true problem in the model. Second, to avoid assumptions regarding the rationality of the decision maker, and thus to compute the certainty closure (the set of all possible solutions) to the model, rather than one postulated solution satisfying some criteria. Third, to guarantee the practicality of the approach by seeking a polynomial algorithm to derive the certainty closure. This last point is essential to allow the user to interact with the system with a reasonable response time.

If the certainty closure is empty, we can infer that the problem is unsatisfiable due to the constraint network and not due to the data. Thus the user can separate data and modelling issues. If the certainty closure is non-empty, we provide support for the user to experiment with the model to see which data refinement has influence on the set of possible solutions. This corresponds to a form of sensitivity analysis. Thus we facilitate interaction by offering not a tool or interface for the user to interact with, but rather reliable information to allow the user to interact as he wishes.

Our purpose, then, is to introduce a framework based on the CSP formalism that can be applied to the heterogeneous constraint classes and computation domains found in real-world LSCOs. Having seen the value of the certainty closure in the network problem, our purpose in this paper is to generalise it by identifying the properties one needs to derive the closure to any LSCO. We model explicitly what is known about the uncertain data (for instance, by an interval of values) in terms of an *uncertain constraint satisfaction problem*. We seek its certainty closure, which provides a correct solution set to the uncertain CSP, excluding no solution possible given the present knowledge of the data. Applications of the certainty closure to other computational domains then follow as instances of the framework. To ensure its practical value in these instances, we show how the closure can be found in an efficient way by a transformation-based approach.

In the following section we develop the certainty closure to the network traffic analysis problem. The concepts and algorithm introduced are generalised in Sect. 3 in a formal presentation of the framework. In Sect. 4 we review and contrast with related work. Sect. 5 concludes the paper with some areas for future investigation.

2 Case Study: Network Traffic Analysis

We present the real-world LSCO where we first defined the certainty closure. The problem is, given a known IP network with incomplete and possibly erroneous traffic measurements at routers, to determine guaranteed bounds for each end-to-end traffic flow [13]. It is known that the true problem must be satisfiable, because the network exists and is executing. The complexity is in adequately handling the data, in guaranteeing the right problem is being solved, and in seeking tight bounds.

The initial approach to the problem was to model it as a CSP, using parameter-based data correction to the uncertain LSCO in order to obtain a consistent, deterministic model. From this model, a hybrid constraint and linear programming method was used to find the bounds. Our aim was to investigate whether the data correction was leading to the true problem, and hence whether the bounds obtained were reliable. We illustrate our experiences on an example fragment of a network.

Initial Model and Method. Consider the fragment of an IP network shown in Fig. 1. Four nodes, corresponding to routers and designated A–D, are shown, together with the bidirectional traffic flow on each link. Each router makes decisions on how to direct traffic it receives, based on a routing algorithm and local flow information.

The network can be modelled as a classical CSP as follows. The variables correspond to the traffic flow between end-points, and their domains are the non-negative reals: $V_{ij} \in \mathbb{R}^+$ is the volume of traffic entering the network at node i and leaving it at node j. The constraints in the network problem form a linear flow model. They state that the volume of traffic through each link in each direction is the sum of the traffic entering the link in that direction. There is also an upper bound (in this case, 64) on the flows that use only a single link, such as V_{ab} and V_{ad}. The data, seen as coefficients in the constraints and denoting the volume of traffic measured at routers, is real and bounded but unknown; it is independent.

Due to the quantity of information required to represent the traffic data exactly, and to the time gap between reading router tables, it is not possible to sample the true traffic in an entire network at one instant. Instead, we measure the aggregated flow volume on a link over a given time interval, and derive the volume of traffic on the link as the difference between end and start measurements. The result is that the data information obtained is inevitably incomplete and erroneous. On the link D→C, for example, the flow might measure as 70 at D and as 80 at C, whereas the true value, equal at both

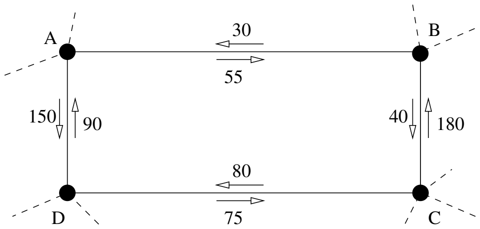

Fig. 1. Traffic flow in a network fragment

nodes, is presumably somewhere between. Classically, we might choose one value in the possible range to work with: the median for instance.

Secondly, when there are two paths of equal cost (from the perspective of the routing algorithm), the traffic is split equally between them in 90% of cases. This is true for flows from A to C, for example. To simplify the model, the initial approach assumed that the traffic is split equally in all such cases.

Supposing these modelling decisions, consider the traffic flow problem:

$$\begin{array}{ll} \texttt{A}\rightarrow\texttt{D} & V_{ad} + 0.5V_{ac} + 0.5V_{bd} = 150 \\ \texttt{D}\rightarrow\texttt{C} & V_{dc} + 0.5V_{db} + 0.5V_{ac} = 75 \\ \texttt{C}\rightarrow\texttt{B} & V_{cb} + 0.5V_{ca} + 0.5V_{db} = 180 \\ \texttt{B}\rightarrow\texttt{A} & V_{ba} + 0.5V_{bd} + 0.5V_{ca} = 30 \end{array}$$

(omitting the four similar traffic constraints in the clockwise direction). The CSP that results is unsatisfiable. The initial approach therefore employed a data correction procedure (minimising deviation on the link volumes) in order to reach a satisfiable, deterministic model.

Certainty Closure Approach. Another common interpretation to the unsatisfiability would be that the problem is over-constrained. But could the lack of solution be due to the assumptions and approximations made? Our approach aimed at answering this question, by making sure the true problem is embedded in the model, and by seeking all its possible solutions without data approximation or decision-making choices.

We did not attempt error correction and, further, decided to model the actual dispatching of traffic, known to be anywhere between 30–70% between two paths. But the more accurate CSP that results is again unsatisfiable. At this point the erroneous approximation to the traffic flow data became clear. We decided to remove all approximations and represent the uncertain flow measurements explicitly. Modelling the problem as an *uncertain CSP* (in which $[\underline{a}, \overline{a}]$ denotes an interval), we now have:

$$\begin{array}{ll} \texttt{A}\rightarrow\texttt{D} & V_{ad} + [0.3, 0.7]V_{ac} + [0.3, 0.7]V_{bd} = [135, 160] \\ \texttt{D}\rightarrow\texttt{C} & V_{dc} + [0.3, 0.7]V_{db} + [0.3, 0.7]V_{ac} = [70, 80] \\ \texttt{C}\rightarrow\texttt{B} & V_{cb} + [0.3, 0.7]V_{ca} + [0.3, 0.7]V_{db} = [180, 190] \\ \texttt{B}\rightarrow\texttt{A} & V_{ba} + [0.3, 0.7]V_{bd} + [0.3, 0.7]V_{ca} = [25, 40] \end{array}$$

The true values for the V_{ij} depend on the true values for the flows, which are unknown. Thus we cannot hope to give a single value assignment and declare it to be 'the' true solution; however we can give an interval for each variable and assert that the solution lies within. Hence the best information we can produce based on the last model is: $V_{ac} \in [0, 150], V_{ad} \in [30, 64], V_{db} \in [32, 200], V_{dc} \in [0, 40], V_{ca} \in [0, 133], V_{cb} \in [17, 64], V_{bd} \in [0, 133]$, and $V_{ba} \in [0, 20]$ (and the four single-link flows in the clockwise direction). The system can tell the user there is a solution to the problem corresponding to at least one possible realisation of the data. If the user informs the system of further knowledge about the data — either observed, or from a domain specialist, or assumptions the user is willing to consider — then the closure can be refined.

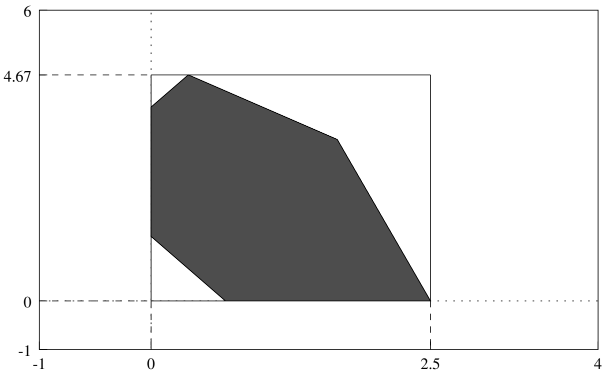

Fig. 2. Tight and certain bounds: the certainty closure is the shaded region

Implementation. The problem modelling and solving uses the ECL^{i}PS^e platform [14], exploiting the `ic` interval computation library. The library provides a *bounded real* datatype: an interval representing an unknown real value; and with safe arithmetic, constraint propagation and search for systems with arbitrary combinations of finite domain and bounded real variables and constraints. Using `ic`, we can model constraints such as `D→C` simply by `Vdc + 0.3__0.7 Vdb + 0.3__0.7 Vac = 70__80`. That is, uncertain coefficients are written naturally and no explicit naming is necessary.

The uncertain CSP is an instance of a *positive orthant interval linear system*. The certainty closure, projected onto the domain of each variable, provides guaranteed intervals. Of the means to calculate the closure — including interval CSP [2] and interval linear programming [6] methods — we chose instead to transform the problem to an equivalent certain CSP. The transformed problem is equivalent in that its complete solution set coincides with the certainty closure of the uncertain CSP. The transformation is illustrated in Example 1. Its advantage over other methods is that both the transformation and the solving of the resulting equivalent problem can be achieved in polynomial time; and that existing solving algorithms can be exploited. For a full description of the transformation, the reader is referred to [21].

Example 1. From the initial system P with constraints $\mathbf{A} \cdot V \, R \, \mathbf{b}$, where

$$\mathbf{A} = \begin{pmatrix} [-2,2] & [1,2] \\ [-2,-1] & -1 \\ 6 & [1.5,3] \end{pmatrix}, \; V = \begin{pmatrix} V_1 \\ V_2 \end{pmatrix}, \; R = \begin{pmatrix} \leq \\ = \\ = \end{pmatrix}, \text{ and } \mathbf{b} = \begin{pmatrix} [3,4] \\ [-5,5] \\ [4,15] \end{pmatrix},$$

the equivalent CSP, $P' = \tau(P)$, we obtain has constraints $A' \cdot V \leq b'$, where

$$A' = \begin{pmatrix} -2 & -2 & 1 & -6 & 6 \\ 1 & -1 & 1 & -3 & 1.5 \end{pmatrix}^\top \text{ and } b' = \begin{pmatrix} 4 & 5 & 5 & -4 & 15 \end{pmatrix}^\top .$$

The transformation works on the bounds of the data values, which suffices by convexity; we will outline it in Sect. 3.4. The increase in the number of constraints (from three in P to five in P') is a result of replacing each equality by a pair of inequalities as a prelude to the transform. The decision variables remain unchanged. □

We efficiently solve the equivalent CSP using Linear Programming (LP). The upper and lower bounds on the possible values for each variable V_i are found by solving two linear programs, with objective $\max V_i$ and $\min V_i$ respectively. Thus with $2n$ simplex iterations we obtain the projection of the certainty closure onto the domain of each variable. For Example 1, this gives $\mathrm{Cl}(P) \subset ([0, 2.5], [0, 4.6667])$, shown in Fig. 2.

Outcome. Compared to the initial approach to the problem, the certainty closure has lead to more reliable quantitative results and to improved understanding of the relationship between network topology and traffic flow. First, we can guarantee the user that the model contains the true problem and thus the result contains all the possible solutions to the true problem. This constitutes a proof showing whether or not previous models and results are accurate. Second, we can give insight into the sources of bottlenecks in the network. In particular, data and modelling issues are be separated. When the interval found for a flow variable is empty, we can be sure that the problem as specified is truly unsatisfiable. Likewise, very tight bounds for a flow indicate that in all possible solutions the traffic behaves in a similar way, while broad bounds indicate a volatile flow; a lower bound on a link close to its capacity indicates a critical, saturated link. The exact values for the flows are less important than the understanding gained of global behaviour in the network.

3 Formal Framework

The idea of a general framework is to take the efficient derivation of the certainty closure introduced in the previous section and generalise it, so that we can apply the method of transformation to an equivalent certain CSP in other computation domains. In this section we formally define the concepts of an uncertain CSP and its certainty closure, and characterise the properties which guarantee a correct transformation.

The case study raises three points about LSCOs with incomplete data knowledge, which the certainty closure framework will address:

- The data in a CSP model is reflected in the domains of the variables *and* in the coefficients in the constraints. Treating the data in the network problem adequately revealed the true reason for the unsatisfiability.
- When the uncertainty is in the knowledge of the data, the user does not desire a solution to an inaccurate model. Thus we seek a closure that derives all possible solutions to a model that contains the true problem.
- The correctness we guarantee provides the user with a firm basis from which to explore the problem. The system must efficiently derive the closure to make such interactive exploration possible.

Each element of the framework has a clear purpose. Firstly, the concept of an *uncertain CSP* (UCSP) aims at completing the CSP-based description of a LSCO by bringing data as a primary concept into the CSP formalism. This allows us to provide a model that contains the true problem, and to provide deeper understanding of the problem structure because data approximation and the modelling issues are not amalgamated.

Secondly, to reason about an uncertain CSP we seek its *certainty closure*, which corresponds to all the information that can be derived accurately about the model. The aim is to ensure correctness by providing the user with the set of all potential solutions to the uncertain problem such that each holds under at least one realisation of the data. There is no assumption of rationality of the decision maker.

Finally, to derive the closure of an uncertain CSP, we give two *resolution forms*, both based on tackling standard CSP models. We propose a general framework to reason on UCSPs without restriction to a class of constraints or data. Therefore, instead of defining a new algorithm, we look for resolution forms that make no requirement regarding the model and thus the solving techniques. Methods from CP or OR can be used as best suits a given computational domain; continuous and discrete problems can be considered without distinction. Core to our contribution is the second resolution form, transformation of an uncertain problem to a classical CSP which can then be solved using existing techniques. We give the properties required by the transformation and solution operators to guarantee the correctness and practicality of the approach.

3.1 Preliminaries

We consider the CSP formalism since it has the generality we desire to model LSCO problems. Recall that 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 $\mathcal{C}$ is a finite set of constraints. A solution is a complete consistent value assignment. This holds, in theory at least, even for non-discrete domains (although operationally the answers differ: e.g. for an interval CSP, a smallest consistent box [2]). On occasion, we will be informal in moving between a full constraint problem and its constraints. For instance, we will speak about the closure of (the conjunction of) a set of constraints, in place of the closure of a UCSP.

Recall further that, with respect to a given computational domain, a constraint domain specifies the syntax and semantics of permitted constraints. It details the constants, functions and constraint relations. The constants we will 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 assume the user has some knowledge of the possible values for the coefficients, or bounds on their range. Call the range of possible values of a coefficient λ_i 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 their exact values are unknown.

Example 2. The constraint $X > 2$ is certain, since the values of all its coefficients (here, just the one) are known. The constraint $X > \{2, 3, 4\}$ is uncertain. Its sole coefficient λ_1 has uncertainty set $U_1 = \{2, 3, 4\}$. □

Regarding the data, following Ben-Tal and Nemirovski [1], a *data realisation* is a fixing of the coefficients to values; in related literature, the terms *possible world* and *context space* are also used. The notation $\widehat{\cdot}$ will denote certainty. For an uncertain CSP P, we will say that any certain CSP $\widehat{P}$, corresponding to a data realisation of the coefficients of P, is a *realised* CSP, and write $\widehat{P} \in P$. Since each uncertain constraint is made certain by a realisation, $\widehat{P} = \langle \mathcal{V}, \mathcal{D}, \widehat{\mathcal{C}} \rangle$, where $\widehat{\mathcal{C}} \in \mathcal{C}$ denotes a set of *realised* constraints. In the same way, a realisation of a constraint c will be denoted $\hat{c} \in c$.

3.2 Uncertain Constraint Satisfaction Problem

The notion of an uncertain constraint satisfaction problem (UCSP) allows us to bring data as a primary concept into constraint programming. An uncertain CSP is a simple extension to a classical CSP with an explicit description of the data.

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 are uncertain. The finite set of coefficients is denoted by* Λ*, and the set of corresponding uncertainty sets by* $\mathcal{U}$*.*

In this paper we will assume the coefficients are independent. The uncertainty set $\mathcal{U}$ is then the Cartesian product of the uncertainty sets of the coefficients, i.e. the Cartesian product of their possible values. Other than this, there is no requirement as to the nature of the data or the representation of $\mathcal{U}$: rather, how the uncertainty should be represented — a set of values, an interval, an ellipsoid, or otherwise — is an issue for each computation domain.[1]

Example 3. The final constraint problem of Sect. 2 is an uncertain CSP. All eight traffic constraints are uncertain, with — due to the data of the application — the uncertainty specified by real intervals. Accordingly, $\mathcal{U}$ is a Cartesian product of intervals. □

Example 4. Let X and Y be variables with integer domains $D_X = D_Y = [1, 5]$. Let c_1 be the simple uncertain constraint: $X > \lambda_1$, and c_2 the certain constraint: $|X - Y| = \lambda_2$. Let λ_1 and λ_2 have uncertainty sets $U_1 = \{2, 3, 4\}$ and $U_2 = 2$ respectively. Then $\langle \mathcal{V}, \mathcal{D}, \Lambda, \mathcal{U}, \mathcal{C} \rangle$ is a UCSP. In contrast to Example 3, the data here is discrete. □

Consider, to help describe the ideas behind the formal framework, the following analogy between solving a standard CSP and reasoning about an uncertain CSP. To reason about the data in a UCSP P, we consider a meta interpretation which is seen by viewing the constraints in P as variables with domain their realisations. An uncertain constraint (variable) is defined in terms of the set of constraint realisations to which it can be instantiated (elements of its domain). The reasoning about a UCSP aims at computing the certainty closure. In the first resolution form, enumeration, this is achieved by generating all the realised CSPs and solving each of them. This corresponds to a simple generate-and-test procedure over the meta problem. The second resolution form is more complex since it aims at transforming the domains of the meta variables into a conjunction of constraints. This means that all the solutions to the UCSP are embedded into the solution set to a single, certain CSP model. In other words, the complete solution set to the transformed problem is the certainty closure of the UCSP.

In order to describe the resolution forms, we must define the algebraic structure over which the ordering relation between uncertain and certain constraints is specified; only then can we reason about uncertain constraints in terms of certain constraints and ensure the correctness and equivalence of the inference produced. The central idea is the observation that uncertain constraints, with suitable operations, form a lattice:

[1] If we restrict to finite domains and discrete data, a UCSP $\langle \mathcal{V}, \mathcal{D}, \Lambda, \mathcal{U}, \mathcal{C} \rangle$ can be viewed as a *mixed CSP* $\langle \Lambda, L, \mathcal{V}, \mathcal{D}, \mathcal{K}, \mathcal{C} \rangle$ with $\mathcal{U}$ being the complete solution set of the CSP $\langle \Lambda, L, \mathcal{K} \rangle$ induced by the parameters. However, as Sect. 4 will discuss, the objectives and algorithms of the two frameworks are quite different.

Proposition 2 (Constraint lattice). *Let $\mathbb{C}$ be the set of all constraints, certain and uncertain, that can arise with respect to a computational domain. With conjunction and disjunction as meet and join, $\mathbb{C}$ is a distributive lattice. With logical implication of constraints, $\mathbb{C}$ has a natural partial order. Let $\widehat{\mathbb{C}} \subset \mathbb{C}$ be the subset of certain constraints; then $\widehat{\mathbb{C}}$ forms a sublattice.* □

This result means that any UCSP P can be described by an element of a suitable constraint lattice: given the domains of the variables and the uncertainty sets of the coefficients, P is described by $\mathcal{C} \in \mathbb{C}$. The same is true for any classical CSP.

Conjunction, disjunction and implication are defined for certain constraints (i.e. on $\widehat{\mathbb{C}}$) in the usual way. Suitable definitions must be given for the extension to $\mathbb{C}$, but how we make the extension depends on what kind of solution we seek. The correctness paradigm entails that an assignment satisfies an uncertain constraint if it satisfies *at least one* realisation. Hence implication is defined by: if every assignment that satisfies some realisation of $c_1 \in \mathbb{C}$ also satisfies some realisation of $c_2 \in \mathbb{C}$, then c_1 implies c_2. Conjunction and disjunction of uncertain constraints are defined in the same manner.

In the next section, we will see that the closure of a UCSP can also be described as an element of $\mathbb{C}$. Thus, firstly and importantly, mappings from $\mathbb{C}$ to itself can encapsulate the solving process. Correctness of the certainty closure will be guaranteed by properties of the mappings. Secondly, knowledge refinement can be seen in terms of a *subsumed-by* order on solutions. Should we learn more about the data, the revised certainty closure will be subsumed by the old.

3.3 Certainty Closure of a UCSP

The certainty closure is the tightest correct solution set to a UCSP. Given the current state of knowledge about the data, *the* certainty closure is the union of all solutions to realisations. Any enclosing outer approximation to this we call *a* certainty closure.

Definition 3 (Certainty closure). *Let P be a UCSP $\langle \mathcal{V}, \mathcal{D}, \Lambda, \mathcal{U}, \mathcal{C} \rangle$. The* certainty closure $\mathrm{Cl}(P)$ *of P is the union of solutions such that: every solution that is feasible under one or more realisations of P is contained in* $\mathrm{Cl}(P)$. *Let S denote a solution satisfying a realised CSP $\langle \mathcal{V}, \mathcal{D}, \widehat{\mathcal{C}} \rangle$. Then we can write the certainty closure as:*

$$\mathrm{Cl}(P) = \bigcup_{\widehat{\mathcal{C}} \in \mathcal{C}} \; \bigcup_{S \text{ satisfies } \widehat{\mathcal{C}}} \{ S \}$$

Example 5. Let P be the UCSP of Example 4, in which the variables $X, Y \in \mathbb{Z}$ have domains $[1, 5]$. The certainty closure is $(X, Y) \in \{(3, 1), (3, 5), (4, 2), (5, 3)\}$. □

If a solution S is possible under *some* realisation of the data, it is part of the certainty closure; if we learn more, it might be possible to exclude it.

Just as a UCSP can be viewed as an element of the constraint lattice $\mathbb{C}$, so can its certainty closure: the closure can be specified as a disjunction of constraints that describe the solutions S. The point of this equivalent, constraint-based view is that we describe the derivation of the certainty closure of a UCSP by an operator (i.e. a mapping) on $\mathbb{C}$. This facilitates a compact proof of correctness, independent of computational domain.

As in the network problem, the *uncertain solution operator* which derives the certainty closure will be based on a transformation from a UCSP P to an equivalent certain problem $\tau(P)$, equivalent in that the set of all solutions to the latter coincides with the certainty closure to the former. To define this notion of equivalence, we need to be able to compare the solution sets of CSPs and UCSPs. We say that a constraint c_2 *subsumes* a constraint c_1 if the set of all solutions of the latter contains that of the former:

Definition 4 (Order). *Recall the subsumed-by partial order on $\widehat{\mathbb{C}}$, defined by Tsang [18].*[2] *Let $\preceq$ be an extension of the order to $\mathbb{C}$ such that $c_2 \in \mathbb{C}$* subsumes *$c_1 \in \mathbb{C}$ if and only if $\mathrm{Cl}(c_2)$ subsumes $\mathrm{Cl}(c_1)$.*

This partial order is well-defined because $\mathrm{Cl}(c)$ is always a certain constraint. As an example, let c_1 be $X > \{2, 3, 4\}$ and $\hat{c}_1$ be $X > 2$. Then c_1 and $\hat{c}_1$ are equivalent under $\preceq$; since, of course, $\hat{c}_1$ is the certainty closure of c_1, this is exactly what we expect.

Representation. We can represent the certainty closure in different ways, depending on whether the user requires the exact closure and on the computational domain of the application. There may be a trade-off between tractability and closeness of approximation of a representation to $\mathrm{Cl}(P)$. One particular representation is to amalgamate the elements described by the closure across the realisations and project onto the variable domains. This gives a closure of possible values for each variable such that, whatever the data realisation, the variables are guaranteed to take values within this closure.

In our example, $\mathrm{Cl}(P)$ projected in this way is: $X \in \{3, 4, 5\}$, $Y \in \{1, 2, 3, 5\}$, or, described by a constraint (recall that the variables have domains $[1, 5]$), $\mathrm{Cl}(P) = (X > 2) \wedge (Y \neq 4)$. For reasons of tractability, namely, to not enumerate all $\widehat{P} \in P$, we might bypass calculation of the exact representation of $\mathrm{Cl}(P)$ and go straight to the tightest enclosing interval (compare with the resolution techniques of generalised propagation [15]), representing the certainty closure as: $X \in [3, 5]$, $Y \in [1, 5]$. Some knowledge regarding the possible values of Y, however, is lost.

3.4 Finding the Certainty Closure

We will look at two resolution forms — two possibilities to move from an uncertain CSP to its certainty closure. The first is to consider every data realisation. Each gives rise to a certain CSP, which we solve, and the closure is then formed from all the solutions to the satisfiable CSPs. The second form is to transform the uncertain CSP. We find and then solve an equivalent certain CSP: the set of all its solutions is the closure to the UCSP. We saw an example of this second form of uncertain solution operator in Sect. 2.

Enumerating Realised CSPs. As an exhaustive technique, enumeration requires operationally there be only finitely-many, $M < \infty$, realisations of the data. Enumeration finds the exact set of all solutions, explicitly showing the relationships among the variable values. However, there are two potential disadvantages. The first is that the cost of

[2] Intuitively, $\hat{c}_1 \in \widehat{\mathbb{C}}$ is *subsumed-by* $\hat{c}_2 \in \widehat{\mathbb{C}}$ if for every satisfying tuple t_1 to $\hat{c}_1$ there exists a satisfying tuple t_2 to $\hat{c}_2$ such that t_1 is a projection of t_2.

the enumeration and solving of M possibly similar CSPs grows with M, which can be exponential in the size of the UCSP. This said, in a given application domain, it may be possible to exploit knowledge of the structure of the realised problems. The second disadvantage is that enumeration is not straight-forward with continuous data; constraints in which it appears must be handled differently.

Solving an Equivalent CSP. A second means to find the certainty closure is by a transformation from the uncertain constraint problem to an equivalent certain problem. The advantage of this resolution form over the first is that it is potentially more tractable and more general. It is more tractable because only one CSP need be solved to find the closure, and more general because continuous data is easily accommodated.

The issues related to this approach are twofold: finding a certain CSP equivalent to the UCSP — one whose set of all solutions coincides with the certainty closure to the original problem — and solving it efficiently. We achieve the first part by seeking a transformation operator from UCSP to CSP which satisfies certain properties; for the second part we can use any existing technique appropriate to the computational domain.

Recall that a classical CSP is solved by propagation and search: one calculates the fixed-point of some local consistency operators and (if necessary) explores the search space. Since we do not restrict ourselves to any solving algorithm, the distinction in solving discrete and continuous CSPs, and the methods used to solve CSPs, are not relevant: the essential point is to guarantee that correct solutions are derived. Hence, we encapsulate fully solving a CSP by a *solution operator*.

We have seen that a CSP $\widehat{P}$ is described by an element of a certain constraint lattice $\widehat{\mathbb{C}}$ and, likewise, so is every solution to $\widehat{P}$. Hence we define a solution operator as a map from $\widehat{\mathbb{C}}$ to itself. A solution operator provides the conjunction of a set of solutions to $\widehat{P}$. The conjunction may be empty, indeed must be if the CSP is inconsistent. A *complete* solution operator is one that yields the set of all solutions to a CSP.

Definition 5 (Solution operator). *Let $\widehat{P} = \langle \mathcal{V}, \mathcal{D}, \mathcal{C} \rangle$ be a certain CSP. Let $\phi : \widehat{\mathbb{C}} \to \widehat{\mathbb{C}}$ be a map such that $\phi(\mathcal{C})$ describes a set of solutions to the CSP. If ϕ obeys:*

1. *Contraction* — *The final state is a subset of the initial state: $\phi(\mathcal{C}) \preceq \mathcal{C}$*
2. *Monotone* — *Subsumed-by order respected: $\mathcal{C}_1 \preceq \mathcal{C}_2 \Longrightarrow \phi(\mathcal{C}_1) \preceq \phi(\mathcal{C}_2)$*
3. *Idempotence* — *Further application of ϕ yields no further solutions*

Then we say that ϕ is a solution operator. *If further $\phi(\mathcal{C})$ describes the set of all solutions to $\widehat{P}$, we say ϕ is* complete *for $\widehat{P}$.*

Example 6. Consider two solution operators for finite domain CSPs. Let ϕ_1 be the map that corresponds to naive backtrack search. If we insist that the whole search tree be explored, then ϕ_1 will give all solutions; this makes it complete. Secondly, let ϕ_2 be the map that corresponds to backtrack search with forward checking. ϕ_2 would be expected to be more efficient than ϕ_1, i.e. find solutions in less time, but aside from operational behaviour, the two operators serve the same purpose for the class of CSPs. □

Similarly, a solution operator for uncertain CSPs is nothing more than a map that yields $\mathrm{Cl}(P)$ when given P. Formally, it is defined as a mapping from $\mathbb{C}$ to $\widehat{\mathbb{C}}$ with certain properties; the last of these is key to reflect the correctness paradigm.

Definition 6 (Uncertain solution operator). *Let P be a UCSP. An* uncertain solution operator *is a map $\rho : \mathbb{C} \to \widehat{\mathbb{C}}$ such that $\rho(\mathcal{C}) = \mathrm{Cl}(P)$. An uncertain solution operator ρ must obey the contraction, monotone and idempotence properties, and in addition admit* correctness under uncertainty*: i.e. $\rho(\hat{c}) \preceq \rho(c) \;\; \forall \hat{c} \in c, c \in \mathbb{C}$.*

Enumeration is one way to build an uncertain solution operator. Another is to use a transformation, which we characterise it as a *certain equivalence transform* (CET):

Definition 7 (Certain Equivalence Transform). *A map $\tau : \mathbb{C} \to \widehat{\mathbb{C}}$ is a* certain equivalence transform *if it: (1) preserves certainty, i.e. $\tau(\hat{c}) = \hat{c} \;\; \forall \hat{c} \in \widehat{\mathbb{C}}$; (2) is a closure operator, i.e. is increasing, monotone and idempotent; and (3) distributes over meet.*

The properties which characterise a certain equivalence transform allow us to guarantee correctness of the uncertain solution operator. Preservation of certainty and the closure properties ensure that a certain constraint system is found. The final property governs the behaviour on conjunctions of constraints. Together, the properties of a CET ensure that the equivalent CSP obtained is such that its complete solution set contains the certainty closure to the original problem. If the converse holds, τ is a *tight* certain equivalence transform. That is, if τ is a tight CET, we need know nothing more about the transform to be sure this resolution form gives the exact closure. If, on the other hand, τ is a non-tight CET, we obtain only an outer approximation to the closure. However, there is often value in such an approximation, if suitably close, since correctness is retained even for non-tight CET.

Proposition 8 sums up the result: an uncertain solution operator ρ can be defined as a composition of a tight CET τ and a complete solution operator ϕ. Think of ρ as first finding an equivalent certain problem, then applying a solution operator to it.

Proposition 8 (Closure by transformation). *If τ is a tight CET and ϕ a solution operator complete for $\tau(\mathcal{C})$, then $\rho = \phi \circ \tau$ is an uncertain solution operator.*

Proof (sketch). The main part of the proof is to show that the set of all solutions to the equivalent CSP coincides with the certainty closure of the original UCSP. From the closure property of τ, we have for all $c \in \mathbb{C}$, $c \preceq \tau(c)$. But $\preceq$ is the subsumed-by order — crudely, the set of all solutions to the LHS is contained in the set of all solutions to the RHS — hence, since $\tau(c)$ is a certain constraint, $\mathrm{Cl}(c) \preceq \phi(\tau(c))$. Since this holds for all $c \in \mathbb{C}$, setting $c = \mathcal{C}$ we have $\mathrm{Cl}(\mathcal{C}) \preceq \phi(\tau(\mathcal{C}))$. The converse, $\phi(\tau(\mathcal{C})) \preceq \mathrm{Cl}(\mathcal{C})$, follows at once from the tightness of τ. Hence, $\rho(P) = \mathrm{Cl}(P)$. □

Example 7. Consider the class of UCSPs with two variables, X and Y with domains in $\mathbb{R}^+$. Suppose the constraints have the form c: $\mathbf{a_1}X + \mathbf{a_2}Y \leq \mathbf{a_3}$, where $\mathbf{a_i} = [\underline{\mathbf{a_i}}, \overline{\mathbf{a_i}}]$ are real, closed intervals. Then a tight CET is given by:

$$\tau(c) = \begin{cases} \underline{\mathbf{a_1}}X + \underline{\mathbf{a_2}}Y \leq \overline{\mathbf{a_3}} & \text{if } \underline{\mathbf{a_2}} \geq 0 \\ \underline{\mathbf{a_1}}X + \overline{\mathbf{a_2}}Y \leq \underline{\mathbf{a_3}} & \text{if } 0 \in \mathbf{a_2} \\ \overline{\mathbf{a_1}}X + \overline{\mathbf{a_2}}Y \leq \underline{\mathbf{a_3}} & \text{if } \overline{\mathbf{a_2}} < 0 \end{cases}$$

This transformation illustrates for a single, simplified constraint the CET used in the uncertain network problem of Sect. 2. For example, the first constraint in the system

P given in Example 1 is transformed from $[-2,2]X + [1,2]Y \leq [3,4]$ to $-2X + Y \leq 4$, since $\underline{\mathbf{a_2}} = 1 \geq 0$. In the application, τ is obtained and proved correct using methods from interval linear programming [6]; τ can be shown to be tight and to have the properties of Definition 7. Our use of LP as the (certain) solution operator ϕ likewise obeys the properties expected in Definition 5 [21]. Hence by Proposition 8, the certainty closure framework guarantees the correctness of the enclosure derived. □

4 Related Work

Existing generic approaches to uncertain data in constraint programming come from quite a different perspective. They propose models and methods to tackle incomplete and dynamic data by seeking robust solutions to the problem. The *mixed CSP* framework [9] of Fargier et. al., defined for discrete data, seeks a solution that holds under the most possible realisations of the data (worlds); the *stochastic CSP* framework [19] of Walsh attaches a probability distribution to parameters and seeks a solution that maximises expectation. The purpose of computing robust solutions is to ensure that whatever the real world situation, the produced solution holds under most cases. This is semantically ideal for dynamic changes but inadequate for handling data errors where one is certainly not looking for a solution that satisfies as many erroneous models as possible. In the context of incomplete and erroneous data, the user expects a guarantee that an accurate model is drawn and correct information is derived from real data, be it discrete or continuous.

The robustness and correctness paradigms lead to semantic and operational differences. Consider the simple uncertain constraint $X > [20, 30]$. Seeking robustness entails satisfying the intersection of all possible hard constraints, i.e. finding an X such that $X > 30$, whereas seeking correctness entails deriving the union of all possible constraints, i.e. the solution set $\{X \mid X > 20\}$, or simply the constraint $X > 20$. Some dedicated research has been carried out to build robust solutions in specific applications domains, particularly in scheduling [7, 16].

In dealing with unsatisfiability, the approach most widely used in CP consists of reasoning at the constraint level. When the model is unsatisfiable, the interpretation usually made is that the problem is over-constrained. The potential data issue is not considered since, indeed, there is no clear concept of data in a standard CSP model. Research has focused on relaxing constraints and setting priorities when solving such CSPs [5, 10]. Aside from uncertainty, others have begun to focus on providing explanation to the user (e.g. [11]). The certainty closure, in enabling us to distinguish failure due to unsatisfiable constraints from that due to inconsistent data, contributes here.

From the point of view of modelling reals using intervals, the field of interval constraint solving, research has looked at building boxes containing all solutions (*completeness*), and to a lesser extent, boxes containing only solution points (*soundness*) [2]. Benhamou and Goualard have sought complete, sound solution sets in the presence of universally quantified variables [3]. Their work, suitable for a single quantified variable and for classes of constraints that can be negated (which excludes equalities), is not however in the context of uncertainty in the data. For systems of quantified inequalities with interval coefficients, some success has been achieved with approximate algorithms [17]. In all this, the solution boxes built are robust and the data is continuous; we are not aware of any work in CP aimed at building correct solution sets in the presence of uncertain data. The closest parallels are Le Provost and Wallace's work on *generalised propagation*, where a constraint approximating all answers to an arbitrary goal is obtained [15], and the meta-solution reasoning of *constructive disjunction* [20].

While our framework is defined with CP modelling in mind, in concept it is more closely related to work in OR and control theory. In OR, semi-definite problems with uncertain data coefficients have been addressed by seeking an enclosing *robust counterpart*[3] [1]. Given unknown coefficients — due to incomplete knowledge or machine errors when handling reals — the solution to the robust counterpart yields a correct solution set to the original problem. Similarly in control theory, convex modelling (of which interval analysis over the reals is a simple instance) is used to obtain a closure guaranteed to contain the true solution [6, 8].

5 Discussion and Future Work

In this paper we investigated how the successes of constraint programming can be extended to real-world problems with uncertainty. We introduced the certainty closure as a generic framework to allow the modelling of data uncertainty in CP, focusing on incomplete and erroneous data. The framework accommodates both discrete and continuous data, and is complementary to stochastic data models. It guarantees a correct closure in that, whatever the true value of the data, the solution to the corresponding realised CSP is contained within the certainty closure. We demonstrated the use of the framework in the application domain of network analysis with uncertain flow measurements, an application that has shown that the certainty closure brings the user insight beyond just knowledge of the solution set.

A formal framework does not suffice unless its application to real LSCOs is practical. Introducing a new modelling perspective, the uncertain CSP, enables us to reason at the meta level about incomplete data, avoiding assumptions about the nature of the data or the computational domain. We derive the correct solution set using a transformation, giving the freedom to make use of specific existing resolution techniques. The user is provided with reliable information from which he can interact with the problem.

Having developed the transformation and proved correctness from its properties, the logical question is, for which computation domains and constraint classes can we derive an efficient transformation? In the case of the widely-studied $(\mathbb{R}^+, \{\leq, =\})$, the transformation is polynomial and thus the class is tractable. Our current research looks at other computation domains, such as temporal constraints in scheduling. For classes where there does not exist a tractable transformation, we look to find one that corresponds to an outer approximation of the certainty closure, seeking to identify cases when the bounds obtained are able to bring insight, even if they are not tight.

Most models with data uncertainty presently assume independence of the data (e.g. [9, 19]). For the certainty closure, assuming independence retains correctness but loses tightness. Future work will include study of how the framework might be extended to

[3] 'robust' here is used to say that the actual value lies in the enclosure without loss of correctness.

account for dependency. We also wish to explore further how the certainty closure can help the user make inference into the structure and behaviour of the problem. Finally, optimisation in uncertain LSCOs remains to be addressed.

Acknowledgements. This work was partially supported by the EPSRC under grant GR/N64373/01. The authors thank P. Brisset and the anonymous reviewers for their helpful comments and suggestions.

References

[1] A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, 23, 1998.

[2] F. Benhamou. Interval constraints. In *Encyclopedia of Optimization*. Kluwer, 2001.

[3] F. Benhamou and F. Goualard. Universally quantified interval constraints. In *CP-2000*.

[4] T. Benoist, E. Bourreau, Y. Caseau, and B. Rottembourg. Towards stochastic constraint programming: A study of online multi-choice knapsack with deadlines. In *Proc. of CP'01*.

[5] S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, and G. Verfaillie. Semiring-based CSPs and valued CSPs: Basic properties and comparison. In *LNCS 1106*. 1996.

[6] J. W. Chinneck and K. Ramadan. Linear programming with interval coefficients. *J. Operational Research Society*, 51(2):209–220, 2000.

[7] A. J. Davenport, C. Gefflot, and J. C. Beck. Slack-based techniques for robust schedules. In *Proc. of CP'01 Workshop on Constraints and Uncertainty*, pages 7–18, 2001.

[8] I. Elishakoff. Convex modeling — a generalization of interval analysis for nonprobabilistic treatment of uncertainty. In *Proc. of the Intl. Workshop on Applications of Interval Computations (APIC'95)*, pages 76–79, El Paso, TX, 1995.

[9] H. Fargier, J. Lang, and T. Schiex. Mixed constraint satisfaction: A framework for decision problems under incomplete knowledge. In *Proc. of AAAI-96*, pages 175–180, 1996.

[10] E. Freuder and R. Wallace. Partial constraint satisfaction. *Artificial Intelligence*, 58, 1992.

[11] E. C. Freuder, C. Likitvivatanavong, and R. J. Wallace. Deriving explanations and implications for constraint satisfaction problems. In *Proc. of CP'01*, pages 585–589, 2001.

[12] C. Gervet, Y. Caseau, and D. Montaut. On refining ill-defined constraint problems: A case study in iterative prototyping. In *Proc. of PACLP'99*, pages 255–275, London, 1999.

[13] C. Gervet and R. Rodošek. RiskWise-2 problem definition. IC–Parc Internal Report, 2000.

[14] IC–Parc. *ECL^{i}PS^e User Manual Version 5.4*, May 2002.

[15] T. Le Provost and M. Wallace. Generalized constraint propagation over the CLP scheme. *J. Logic Programming*, 16(3):319–359, 1993.

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

[17] S. Ratschan. Uncertainty propagation in heterogeneous algebras for approximate quantified constraint solving. *J. Universal Computer Science*, 6(9), 2000.

[18] E. Tsang. *Foundations of Constraint Satisfaction*. Academic Press, 1993.

[19] T. Walsh. Stochastic constraint programming. In *Proc. of AAAI'01 Fall Symposium on Using Uncertainty within Computation*, pages 129–135, Cape Cod, MA, Nov. 2001.

[20] J. Würtz and T. Müller. Constructive disjunction revisited. In *Proc. of the Twentieth German Annual Conference on Artificial Intelligence (KI-96)*, 1996.

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