Probabilistic Continuous Constraint Satisfaction Problems

Elsa Carvalho Jorge Cruz
Pedro Barahona
Centro de Inteligência Artificial, Universidade Nova de Lisboa, Portugal
elsac@uma.pt, {jc,pb}@di.fct.unl.pt

Abstract

Constraint programming has been used in many applications where uncertainty arises to model safe reasoning. The goal of constraint propagation is to propagate intervals of uncertainty among the variables of the problem, thus only eliminating values that assuredly do not belong to any solution. However, to play safe, these intervals may be very wide and lead to poor propagation. In this paper we present a framework for probabilistic constraint solving that assumes that uncertain values are not all equally likely. Hence, in addition to initial intervals, a priori *probability distributions (within these intervals) are defined and propagated through the constraints. This provides* a posteriori *conditional probabilities for the variables values, thus enabling the user to select the most likely scenarios.*

1 Introduction

Constraint programming is adequate to study many types of systems that arise in engineering [1], biomedical [7] and other types of applications [11], namely dealing with continuous domains. Here, a system is modeled by a number of relations (equality and inequality constraints) between the system's input and output, via the system parameters. The fact that constraint programming is based on relations rather than functions enables the use of the same models in a number of different model-based tasks, namely diagnosis (given inputs and outputs find the value of the system parameters), prognosis (given inputs and system parameters find the value of the output), or testing (given the system parameters find values for the inputs that lead to some intended outputs).

Uncertainty arises naturally in these problems. It may be accommodated by considering that uncertainty in variables and relations is modeled by variables whose domains range over the intervals of possible values. This leads to safe reasoning about systems, namely in continuous domains - intervals represent all values that some variables can take. The role of constraint propagation is to obtain intervals for the other variables, as narrow as possible, thus decreasing the associated uncertainty. However, safe reasoning is not always appropriate. To play safe, wide intervals may have to be assumed, even if the more likely values are included in much smaller intervals. Nevertheless classical constraint programming assumes all values in an interval to be equally likely, contrary to what is expected.

In this paper we introduce probabilistic reasoning in constraint programming to deal with variables for which some probability distributions can be defined *a priori*. The role of constraint programming is then extended to propagate, not only the intervals but also these distributions to other variables so that conditional distributions are obtained from the evidence provided.

The framework has already been successfully applied to inverse problems [5] (which are similar to model-based diagnosis, but where there are no input variables) where uncertainty in output variables, caused by measuring errors and modeled by normal distributions, are propagated and result not only in safe values for the system parameters but also in probability distributions that enable the identification of the more likely values for the system parameters.

Hence, rather than showing applications, the paper aims at formalizing the reasoning and is organized as follows. Sections 2 and 3 formalize, respectively, continuous constraint satisfaction problems and probabilistic models. Then, Section 4 introduces probabilistic continuous constraint solving to compute *a posteriori* probabilities of scenarios given some *a priori* probability distributions and constraints between the variables. This section also provides algorithms to compute those distributions and illustrates the approach with some examples. In Section 5 related work is addressed and compared with the proposed framework. Finally, conclusions and ideas for future work are discussed.

2 Continuous CSPs

A constraint specifies a relation between its variables. Mathematical constraints are precise specifiable relations

among variables, each ranging over a given domain, and are a natural way for expressing regularities upon the underlying real-world systems and their mathematical abstraction. A Continuous Constraint Satisfaction Problem (CCSP) is defined by a set of variables each with an associated real domain of possible values and a set of constraints on subsets of the variables. A constraint specifies which values from the domains of its variables are compatible. A solution to the CCSP is an assignment of values to all its variables, which satisfies all the constraints. More formally the following general definitions are adapted from other authors [15, 2, 19].

Definition 1 (Constraint)**.** *A constraint c is a pair (s, ρ), where s (the constraint scope) is a tuple of m variables $\langle x_1, x_2, \ldots, x_m \rangle$ and ρ (the constraint relation) is a subset of the Cartesian product $D_1 \times D_2 \times \cdots \times D_m$ with D_i representing the domain of variable x_i:*

$$\rho \subseteq \{\langle d_1, d_2, \ldots, d_m \rangle \mid d_1 \in D_1, d_2 \in D_2, \ldots, d_m \in D_m\}$$

Definition 2 (Continuous Constraint Satisfaction Problem)**.** *A CCSP is a triple (X,D,C) where X is a tuple of n real variables $\langle x_1, x_2, \ldots, x_n \rangle$, D is the Cartesian product of their domains $I_1 \times I_2 \times \cdots \times I_n$, with each variable x_i ranging over the real interval I_i, and C is a finite set of numerical constraints expressed as (possibly) non linear equations or inequalities on subsets of the variables in X.*

In the following, whenever d is a tuple of elements associated with all the CCSP variables and s is a subset of such variables, $d[s]$ represents the projection of d with respect to s (only the elements associated with variables from s are in $d[s]$).

Definition 3 (Solution)**.** *A solution of the CCSP (X,D,C) is a tuple $d \in D$ that satisfies each constraint in C, that is:*

$$\forall_{(s,\rho) \in C}\ d[s] \in \rho$$

Whereas in some CCSPs it is important to determine individual solutions, in many practical situations, due to the continuous nature of such solutions, the ultimate goal is to characterize the complete set of solutions.

Definition 4 (Solution Space)**.** *The solution space of the CCSP (X,D,C) is the set $SS \subseteq D$ of all solutions of the CCSP:*

$$\forall_{d \in SS}\ \forall_{(s,\rho)} \in C\ d[s] \in \rho \qquad \textit{(only solutions inside)}$$
$$\forall_{d \in D}\ d \notin SS \Rightarrow \exists_{(s,\rho) \in C}\ d[s] \notin \rho \textit{(no solutions outside)}$$

For illustration purposes consider a CCSP with variables $X = \langle x, y \rangle$, domains $D = [-4, 4] \times [-4, 4]$ and the constraint $C = \{C_1 : y = x + [-1, 1]\}$. Examples of solutions are the tuples $\langle 1, 1 \rangle$, $\langle 0, 1 \rangle$ and $\langle -0.5, -1 \rangle$. The black area in figure 1*a* represents the complete solution space of that CCSP.

Constraint reasoning aims at eliminating value combinations from the initial domains (the initial search space) that do not satisfy the constraints. Usually, during constraint reasoning in continuous domains, the search space is maintained as a set of boxes (Cartesian product of intervals) which are pruned and subdivided until a stopping criterion is satisfied. Pruning is accomplished by eliminating boxes that can be proved inconsistent, that is, do not contain solutions because there is some constraint that cannot be satisfied.

Definition 5 (Consistency)**.** *A set $R \subseteq D$ is consistent with CCSP (X,D,C) iff it contains at least one solution (otherwise it is inconsistent):*

$$\exists_{d \in R}\ \forall_{(s,\rho) \in C}\ d[s] \in \rho$$

In order to eliminate value combinations incompatible with a particular constraint, safe narrowing functions (mappings between boxes) must be associated with the constraint. Efficient methods from interval analysis (e.g. the interval Newton [16]) are often used to implement efficient narrowing functions which are correct (do not eliminate solutions) and contracting (the obtained box is contained in the original).

Definition 6 (Narrowing Function)**.** *Let (X,D,C) be a CCSP. A narrowing function NF associated with a constraint $(s, \rho) \in C$ is a mapping between subsets of D where $\forall_{R \subseteq D}$:*

$$NF(R) \subseteq R \qquad \textit{(contractance)}$$
$$\forall_{d \in R}\ d \notin NF(R) \Rightarrow d[s] \notin \rho \qquad \textit{(correctness)}$$

Once a set of narrowing functions is associated with the constraints of the CCSP, the pruning of a box can be achieved through constraint propagation. Narrowing functions associated with a constraint eliminate some incompatible values from the domain of its variables and this information is propagated to all constraints with that variables in their scopes. The process terminates when a fixed point is attained i.e., the domains can not be further reduced by any narrowing function. It can be proved [17] that the propagation algorithm is correct and terminates independently from the order of application of the narrowing functions during the process. Moreover, if the narrowing functions are monotonic then the propagation algorithm is confluent (the result is independent from their order of application) and computes the greatest common fixed point included in the initial domains.

Definition 7 (Constraint Propagation Algorithm)**.** *Let (X, D, C) be a CCSP. Let set S_{NF} be a set of narrowing functions obtained from the set of constraints C. The constraint propagation algorithm CPA defines a mapping between subsets of D where $\forall_{R \subseteq D}$:*

$CPA(R) \subseteq R$ *(contractance)*
$\forall_{d \in R}\, d \notin CPA(R) \Rightarrow \exists_{(s,\rho) \in C}\, d[s] \notin \rho$ *(correctness)*
$\forall_{NF \in S_{NF}}\, NF(CPA(R)) = CPA(R)$ *(fixed point)*

The pruning achieved through the constraint propagation algorithm is highly dependent on the ability of the narrowing functions for discarding value combinations that are inconsistent with the respective constraint [6]. To obtain further pruning, it is necessary to split the box domains and reapply constraint propagation to each sub-box. In general, continuous constraint reasoning is based on such a branch and prune process which will eventually terminate due to the imposition of conditions on the branching process (e.g. "small enough" boxes are not considered for branching). Nevertheless, since no solution is lost during the process, constraint reasoning provides a safe method for computing an enclosure of the solution space of a CCSP.

Definition 8 (Constraint Reasoning)**.** *Constraint reasoning provides a safe method for computing an enclosure R of the solution space SS of a CCSP (X,D,C), that is:*

$$SS \subseteq R \subseteq D$$

Consider again the CCSP presented earlier. Figure 1*b* represents the enclosure of the solution space of that CCSP, obtained by constraint reasoning. To illustrate the approximative nature of such enclosure a high granularity is considered, i.e., branching is not allowed when the largest domain of a box is smaller than 0.5.

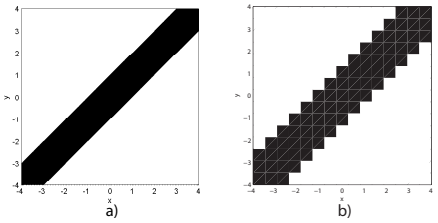

Figure 1. *CCSP* $(\langle x, y \rangle, [-4, 4] \times [-4, 4], \{y = x + [-1, 1]\})$**. (a) SS; (b) SS enclosure.**

When modeling a real world problem using the classical CCSP framework, the uncertainty associated with the problem is modeled by using intervals to represent the domains of the variables. By approximating the solution space this uncertainty is reduced to a minimum. Indeed, the constraint reasoning can be interpreted as a way of reducing uncertainty, by *reshaping* the search space to become a good approximation of the solution space.

However, safe reasoning may be useless, intervals are often very wide, and subsequent constraint propagation is not able to narrow them. In fact, an uncertain value may range over a wide interval but a much narrower interval may include the most likely values. In some problems, the plausibility distribution of values within the bounds of an uncertain parameter, is also known. For instance, uncertainty due to measuring errors may be naturally associated with an error distribution. However, the traditional CCSP framework cannot accommodate such information and thus, for each variable, all values in its domain are considered equally plausible.

3 Continuous Probabilistic Models

Probability provides a classical model for dealing with uncertainty [10]. The basic element of probability theory is the random variable, which plays a similar role to that of the CCSP variables. Each continuous random variable has a domain where it can assume real values. A possible world, or atomic event, is an assignment of values to all the variables of the model. An event is a set of possible worlds. The complete set of all possible worlds in the model is the sample space. If all the random variables are continuous, the sample space is the hyperspace obtained by the Cartesian product of the variable domains, and the possible worlds and events are, respectively, points and regions from such hyperspace. This will be presented more formally using the following definitions (similar definitions can be found in [12, 14, 9]).

Probability measures may be associated with possible worlds or events. A probabilistic model is an encoding of probabilistic information, allowing to compute the probability of any event, in accordance to the axioms of probability.

Definition 9 (Probability Space)**.** *A probability space is a triple* $(\Omega, \mathcal{B}, P)$ *where* Ω *is a nonempty set,* $\mathcal{B}$ *a collection of subsets of* Ω *(that contains the empty set* $\varnothing$ *and is closed under the formation of complements and of finite or countable unions of its members) and P a nonnegative function defined on* $\mathcal{B}$ *such that:*

$$P(\Omega) = 1 \quad and \quad P\Big(\bigcup_{n=1}^{\infty} A_n\Big) = \sum_{n=1}^{\infty} P(A_n)$$

provided $A_n \in \mathcal{B}$ *and* $A_n \cap A_m = \varnothing$ *for each* $n \neq m$*. Function P is a probability measure and the sets belonging to* $\mathcal{B}$ *are called events.*

Definition 10 (Random Variables)**.** *Given a probability space* $(\Omega, \mathcal{B}, P)$*, a random variable is a real-valued function X on* Ω *such that, for all real numbers r:*

$$\big\{\omega \in \Omega : X(\omega) \leq r\big\} \in \mathcal{B}$$

In the continuous case, an assignment of a probability to a point, is representative of the likelihood in its neighborhood. Methods for specifying a probabilistic model employ,

either explicitly or implicitly, a full joint probability distribution, which assigns a probability measure to each possible world.

Definition 11 (Joint Distributions and Density Functions)**.** *Let $X_1, X_2, \ldots, X_n$ be random variables on the probability space $(\Omega, \mathcal{B}, P)$. Their joint distribution function is defined by:*

$$F(r_1,\ldots,r_n) = P\big(\{\omega \in \Omega : X_1(\omega) \leq r_1, \ldots, X_n(\omega) \leq r_n\}\big)$$

If there is a function f such that:

$$F(r_1,\ldots,r_n) = \int_{-\infty}^{r_1} \cdots \int_{-\infty}^{r_n} f(u_1,\ldots,u_n) du_n \ldots du_1$$

then f is their joint probability density function (p.d.f.).

In the particular case where the random variables are independent their joint density may be computed from their individual densities.

Definition 12 (Independent Random Variables)**.** *If $X_1, X_2, \ldots, X_n$ are independent random variables on the probability space $(\Omega, \mathcal{B}, P)$ and have respective p.d.f.s $f_1, f_2, \ldots, f_n$ then their joint p.d.f. is the product of the individual p.d.f.s:*

$$F(r_1,\ldots,r_n) = \int_{-\infty}^{r_1} \cdots \int_{-\infty}^{r_n} f_1(u_1)\ldots f_n(u_n) du_n \ldots du_1$$

Probabilistic reasoning [18] aims at incorporating new information, known as evidence, by updating an *a priori* probability into an *a posteriori* probability given the evidence. The *a priori* probability is a description of what is known in the absence of the evidence. For incorporating this evidence, conditioning is used. Conditional probability $P(A|B)$ is the probability of some event A, given the occurrence of some other event B. The *a posteriori* probability is the conditional probability when the relevant evidence is taken into account.

Definition 13 (Conditional Probability)**.** *Given a probability space $(\Omega, \mathcal{B}, P)$, if $A \in \mathcal{B}$ and $B \in \mathcal{B}$ are two events with $P(B) > 0$ the conditional probability of A given B is written $P(A|B)$ and defined as:*

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Definition 14 (Conditional Probability Space)**.** *Given a probability space $(\Omega, \mathcal{B}, P)$, let $B \in \mathcal{B}$ be some event with $P(B) > 0$. With the same Ω and $\mathcal{B}$, define the function Q on $\mathcal{B}$ by:*

$$Q(A) = \frac{P(A \cap B)}{P(B)}.$$

Then $(\Omega, \mathcal{B}, Q)$ is also a probability space.

Definition 15 (Probabilistic Reasoning)**.** *Given a probability space $(\Omega, \mathcal{B}, P)$, and some evidence $B \subseteq \Omega$ probabilistic reasoning provides a method for computing the a posteriori probability space $(\Omega, \mathcal{B}, Q)$ given the evidence.*

Consider a probability space where $\Omega = \mathbb{R}^2$ characterized by a given *a priori* bivariate Gaussian distribution. Consider as *evidence* the event defined by the set of points that satisfy the equation $y = x + [-1, 1]$. Figure 2*a* presents the *a priori* probability space (the evidence is not considered) and figure 2*b* the *a posteriori* probability space, i.e., the conditional probability space given the evidence.

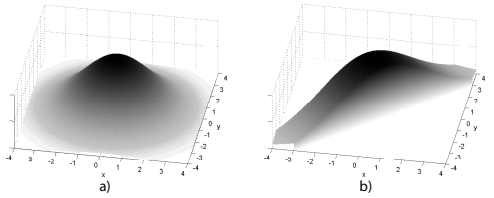

Figure 2. (a) a priori bivariate Gaussian distributed probability space; (b) a posteriori probability space given the evidence $y = x + [-1, 1]$.

4 Probabilistic CCSPs

In the present paper the authors propose the Probabilistic Continuous Constraint Satisfaction Problem (PCCSP) as an extension of a CCSP (preliminary work on this subject was presented in [4]).

Definition 16 (Probabilistic CCSP)**.** *A PCCSP is a tuple (X,D,C,f) where X is a tuple of n real variables $\langle x_1, x_2, \ldots, x_n \rangle$, D is the Cartesian product of their domains $I_1 \times I_2 \times \cdots \times I_n$, with each variable x_i ranging over the real interval I_i, C is a finite set of numerical constraints expressed as (possibly) non linear equations or inequalities on subsets of the variables in X, and f is a non-negative point function defined in D such that:*

$$\int_{I_1} \cdots \int_{I_n} f(u_1,\ldots,u_n) du_n \ldots du_1 = 1$$

In the process of reducing the uncertainty of a PCCSP, there is a combination of continuous constraint reasoning and probabilistic reasoning. While the first reduces uncertainty by *reshaping* the search space, the second *redefines* the search space *a priori* probability distribution by computing an *a posteriori* distribution, based on the constraint reasoning outcome.

Definition 17 (*a priori* Constrained Probability Space)**.** *The a priori constrained probabilistic space of a PCCSP (X,D,C,f) is the probability space* $(D, 2^D, P)$ *where P is a nonnegative function defined on* 2^D *such that:*

$$P(A) = \int \dots \int_A f(u_1, \dots, u_n) du_n \dots du_1$$

For each variable x_i *in X there is a random variable* X_i *on D defined as:*

$$\forall_{d \in D}\ X_i(d) = d[x_i]$$

The joint distribution function for the $X_1, X_2, \dots, X_n$ *random variables is:*

$$\begin{aligned} F(r_1,\dots,r_n) &= P\big(\{d \in D : d[x_1] \leq r_1, \dots, d[x_n] \leq r_n\}\big) \\ &= \int_{\inf(I_1)}^{r_1} \dots \int_{\inf(I_n)}^{r_n} f(u_1,\dots,u_n) du_n \dots du_1 \end{aligned}$$

where f is their joint p.d.f. and $D = I_1 \times I_2 \times \dots \times I_n$*.*

The constraints are the new information that is incorporated in the probabilistic model. The solution space is the event containing all possible worlds that satisfy the constraints. Through constraint reasoning an enclosure of the solution space is obtained. Therefore the *a posteriori* probability is computed as a conditional probability, given the evidence represented by the solution space approximation. This probability is calculated by the conditional probability rule $P(A|SS) = P(A \cap SS)/P(SS)$. The probability of region A given the evidence, is the probability of the subregion of A contained in the solution space approximation, divided by a normalizing factor.

Definition 18 (Solution Space)**.** *The solution space of a PCCSP (X,D,C,f) is the solution space of the CCSP (X,D,C).*

Definition 19 (*a posteriori* Constrained Probability Space)**.** *Given a PCCSP (X,D,C,f) with a nonempty solution space SS and an a priori constrained probabilistic space* $(D, 2^D, P)$ *where* $P(SS) > 0$*, the a posteriori constrained probabilistic space is the probability space* $(D, 2^D, Q)$ *where Q is a nonnegative function defined on* 2^D *such that:*

$$Q(A) = \frac{P(A \cap SS)}{P(SS)}$$

The proposed PCCSP model presents three properties that are in accordance with probability theory. Firstly, in the *a posteriori* constrained probabilistic space of a PCCSP the probability of an inconsistent region is zero.

Property 1. *Given a PCCSP (X,D,C,f) with a solution space SS, an a priori constrained probabilistic space* $(D, 2^D, P)$ *and an a posteriori constrained probabilistic space* $(D, 2^D, Q)$ *then* $\forall_{A \subset D}\ (A \cap SS = \varnothing \Rightarrow Q(A) = 0)$.

Proof. By definition 19,

$$Q(A) = \frac{P(A \cap SS)}{P(SS)} = \frac{P(\varnothing)}{P(SS)} = \frac{0}{P(SS)} = 0 \qquad \square$$

Secondly, the ratios between *a priori* probabilities of regions totally contained in the solution space, are maintained in their *a posteriori* probabilities.

Property 2. *Given a PCCSP (X,D,C,f) with a solution space SS, an a priori constrained probabilistic space* $(D, 2^D, P)$ *and an a posteriori constrained probabilistic space* $(D, 2^D, Q)$*, then:*

$$\forall_{A,B \subset SS} \left(P(B) > 0 \Rightarrow \frac{P(A)}{P(B)} = \frac{Q(A)}{Q(B)} \right)$$

Proof. By definition 19 and the conditions assumed in property 2,
$Q(A) = \frac{P(A \cap SS)}{P(SS)} = \frac{P(A)}{P(SS)}$, similarly, $Q(B) = \frac{P(B)}{P(SS)}$.
Since $P(B) > 0$ then $P(SS) \geq P(B) > 0$ and the result follows. $\square$

Thirdly, the constrained probabilistic space is not changed if the constraint relations are already expressed in the *a priori* constrained probabilistic space.

Property 3. *Given a PCCSP (X,D,C,f) with a solution space SS, an a priori constrained probabilistic space* $(D, 2^D, P)$ *and an a posteriori constrained probabilistic space* $(D, 2^D, Q)$*, then:*
$\forall_{A \subset D}(A \cap SS = \varnothing \Rightarrow P(A) = 0) \Rightarrow \forall_{B \subset D}(P(B) = Q(B))$

Proof. If the condition of property 3 holds for any $A \subset D$ it holds for $A = \overline{SS}$ and so $P(\overline{SS}) = 0$ and $P(SS) = 1$. For any $B \subset D$ and by definition 19 $Q(B) = P(B \cap SS)$. In general $P(B) = P(B \cap SS) + P(B \cap \overline{SS})$ and so $P(B) = Q(B) + P(B \cap \overline{SS})$. Since $(B \cap \overline{SS}) \cap SS = \varnothing$ then $P(B \cap \overline{SS}) = 0$ and $P(B) = Q(B)$. $\square$

4.1 Implementation

As discussed in section 2 (see definition 8), constraint reasoning provides a method for computing a safe enclosure of the solution space (SS) of a CCSP. However, if some regions of the search space were not pruned during constraint reasoning they may contain solutions, although there is no guaranty that they do. In fact, there is no knowledge why such regions are maintained. Was it due to lack of further exploration of this regions or did they contain solutions? Normally, the process of constraint reasoning, leads to non uniform sizes of the boxes that represent the SS approximation. Nevertheless, for reasoning with probabilistic information, some kind of fairness in the exploration of the search space must be guaranteed, so that the obtained *a posteriori* distribution is not biased by heterogeneous search.

In the present PCCSP implementation the stopping criterion is based on the maximum width of the intervals that constitute a box. When all the intervals of a box are smaller or equal to the maximum width allowed ε, that box is no further explored. When all the boxes meet this criterion the search stops. The stopping criterion assures some uniformity of the SS approximation. However, some heterogeneity is still present due to the narrowing ability of any consistency enforcement algorithm, which differs between distinct regions of the search space.

To maintain a generic non parametric representation of the *a posteriori* marginal p.d.f.s, some kind of discretization must be assumed. This is achieved by considering an hypergrid of width ε, i.e., a grid where the dimension is the number of variables in the PCCSP, and each grid unit has width ε in all dimensions. The hypergrid allows to transform the non uniform SS approximation, resulting from constraint reasoning, in a uniform one, providing a fair computation of the marginal p.d.f.s. This is done by overlaying the hypergrid upon the SS approximation, enforcing a *snap to grid* to this region. The new approximation is the set of grid hypercubes that intersect with the original one.

The sequence of images in figure 3 illustrates the described probabilistic constraint reasoning process.

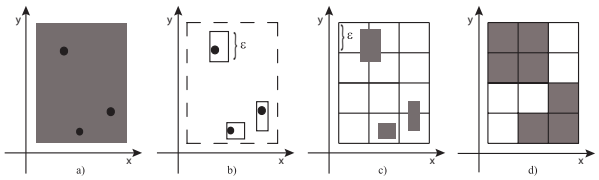

Figure 3. Probabilistic constraint reasoning. (a) Initial search space and SS; (b) SS approximation; (c) Hypergrid; (d) Snap to grid.

Probabilistic constraint reasoning is achieved by function *ssEnclosure*, defined in algorithm 1. Given the CCSP (X, D, C), this function returns the set of ε-hypergrid boxes representing its SS approximation. The auxiliary function *solve* computes the set of ε-hypergrid boxes recursively in H. It begins with the empty set H and the initial search space box D. Then, using the constraint propagation algorithm (*CPA*) it narrows the given box A (line 1). Unless the resulting box is eliminated (line 2) two things may happen. If the obtained box is sufficiently small (line 3) it is decomposed in its respective ε-hypergrid boxes, which contribute to the SS approximation H (line 4). The decomposition is computed by function $\varepsilon Boxes$ following the *snap to grid* process, illustrated in figure 3 *c* and *d*. Otherwise, the obtained box is split in a left and a right box (line 6), which are then subject to the same process (lines 7 and 8).

Algorithm 1 Solution space enclosure

function $ssEnclosure((X, D, C), \varepsilon)$
  $H \leftarrow \oslash$
  $solve(D, H, \varepsilon)$
  return H
function $solve(A, H, \varepsilon)$
1: $A \leftarrow CPA(A)$
2: **if** $A \neq \oslash$ **then**
3: **if** $A.width \leq \varepsilon$ **then**
4: $H \leftarrow H \cup \varepsilon Boxes(A)$
5: **else**
6: $(L, R) \leftarrow split(A)$
7: $solve(L, H, \varepsilon)$
8: $solve(R, H, \varepsilon)$
9: **end if**
10: **end if**

Algorithm 1 computes a finite set of ε-hypergrid boxes that constitutes a safe SS approximation of the PCCSP.

Property 4. *Given a PCCSP (X,D,C,f) with a solution space SS, function ssEnclosure computes a set* $H = \{A_1, \ldots, A_k\}$*, of* ε*-hypergrid boxes* A_j*, where:*

$$SS \subseteq \bigcup_{j=1}^{k} A_j \subseteq D$$

Moreover, the SS approximation computed by algorithm 1 converges to the actual SS of the PCCSP when the size of the ε-hypergrid boxes tend to 0 (assuming that the constraint propagation algorithm detects any inconsistent single value combination).

Property 5. *Given a PCCSP (X,D,C,f) with a solution space SS, if* $\forall_{d \in D}(d \notin SS \Rightarrow CPA(d) = \varnothing)$ *then:*

$$\lim_{\varepsilon \to 0} ssEnclosure((X, D, C), \varepsilon) = SS$$

Once obtained the SS approximation as a set H of ε-hypergrid boxes, algorithm 2 calculates and returns (line 12) the marginal *a posteriori* p.d.f. of a subset of m variables from the PCCSP, discretized according to the ε-hypergrid. For this purpose the algorithm maintains an m-dimensional matrix M, where each dimension corresponds to a variable. In the algorithm, $H_{hull}[x_i]$ (and $A[x_i]$) is the interval obtained by projecting box H_{hull} (and A) into variable x_i. Given two intervals $I_1 = [l_1, r_1]$ and $I_2 = [l_2, r_2]$, the union hull $I_1 \uplus I_2$ is the interval $[min(l_1, l_2), max(r_1, r_2)]$. H_{hull} is a box where each interval is the union hull of the respective intervals of all the boxes in H, i.e. is the smallest box enclosing all the boxes in H (line 2). The length of each dimension l_i is the number of ε segments in which the corresponding variable domain can be divided (line 3). Each

matrix cell thus obtained is initialized to zero (line 4) and its probability value is computed by summing up the contribution of all hypercubes (boxes) that are aligned with that cell (line 5-8), normalized by the sum of all hypercube contributions (lines 1, 9, 11).

Algorithm 2 Calculates marginal *a posteriori* p.d.f.

function $marginal(H, \langle x_1,\ldots,x_m\rangle, P, \varepsilon)$

1: $H_{prob} \leftarrow 0$
2: $H_{hull} \leftarrow \uplus H$
3: $\forall_{1\leq i\leq m} \quad l_i \leftarrow H_{hull}[x_i].width/\varepsilon$
4: $\forall_{1\leq i_1<l_1}\ldots\forall_{1\leq i_m<l_m} \quad M[i_1]\ldots[i_m] \leftarrow 0$
5: **while** $H \neq \oslash$ **do**
6: $\quad A \leftarrow remove(H)$
7: $\quad \forall_{1\leq i\leq m} \quad j_i \leftarrow (A[x_i].left - H_{hull}[x_i].left)/\varepsilon$
8: $\quad M[j_1]\ldots[j_m] \leftarrow M[j_1]\ldots[j_m] + P(A)$
9: $\quad H_{prob} \leftarrow H_{prob} + P(A)$
10: **end while**
11: $\forall_{1\leq i_1<l_1}\ldots\forall_{1\leq i_m<l_m} M[i_1]\ldots[i_m] \leftarrow M[i_1]\ldots[i_m]/H_{prob}$
12: **return** M

4.2 Examples

Consider a CCSP with variables $X = \langle x, y, r, s\rangle$, domains $D = [-4,4]\times[-4,4]\times[2,4]\times[-1,1]$ and constraints $C1 : y = x + s$ and $C2 : x^2 + y^2 = r^2$. Figure 4 represents enclosures of the solution space (projected over x and y) of several variations of that CCSP, obtained by constraint propagation with $\varepsilon = 0.01$. In *a* no constraints are considered. In *b*, *c* and *d* constraints C_1, C_2 and both C_1 and C_2 are, respectively, considered.

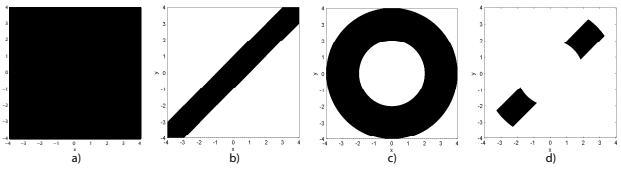

Figure 4. CCSPs SS approximations projected over x and y (a) No constraints; (b) C1; (c) C2; (d) C1 and C2.

Using the PCCSP framework and considering Gaussian distributions for all the variables of the described problem, figure 5 presents the enclosures of the solution space (projected over x and y) of the same variations of that problem, characterized by their respective *a posteriori* distributions. The framework allows the distinction between different regions of the solution space approximation. In situations with a large solution space area this extra information can be useful for choosing a smaller region of interest.

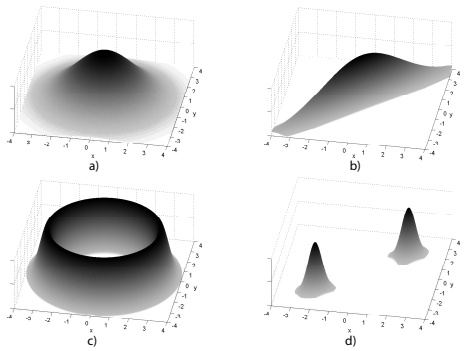

Figure 5. PCCSPs SS approximations projected over x and y (a) No constraints; (b) C1; (c) C2; (d) C1 and C2.

The sequence of images in figure 6 shows that the PCCSP framework agrees with probability theory, namely with its central limit theorem. This theorem states that the sum of a large number of independent and identically-distributed random variables will be approximately Gaussian distributed. Consider the PCCSP defined by the constraints $C_1 : x = \sum_{i=1}^{n} x_i$ and $C_2 : y = \sum_{i=1}^{n} y_i$, where every x_i and y_i are *a priori* uniformly distributed in $[0, 1]$. Figure 6 presents the *a posteriori* distributions projected over x and y (initially unbounded and *a priori* uniformly distributed). In *a* and *b* is shown, respectively, the contour and surface views of the SS approximation for $n = 2$. In *c* and *d* the same views are shown for $n = 4$.

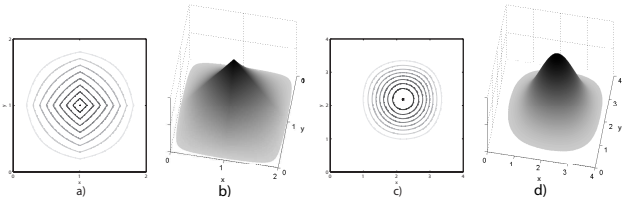

Figure 6. PCCSP SS approximation projected over x and y. Contour and surface views (a) and (b) for $n = 2$; (c) and (d) for $n = 4$.

As expected, for $n = 2$ (fig. 6 *a* and *b*) triangular distributions are obtained. For $n = 4$ (fig. 6 *c* and *d*) the results are already closer to the bell shaped Gaussian distribution illustrated in figure 2*a*.

5 Related Work

A combination of probabilistic and interval representations of uncertainty appears in [13], which uses interval do-

mains to represent the ranges of possible values and allows the incorporation of extra information about their probabilities. Such framework uses interval computations instead of the broader paradigm of continuous constraint reasoning, which makes it less general than the proposed framework.

There are a number of works that aim to combine probabilities and constraint programming or to represent the uncertainty associated with the modeled problems. For example, in [20], there are probabilistic preferences on the values assigned to variables. Other example is [8], called probabilistic constraint satisfaction, where each constraint as a certain probability of being part of the problem. In [3], both valued and semi-ring based constraint satisfaction are proposed, where a value is associated, respectively, with each constraint or with each tuple in a constraint. In [21], a stochastic constraint programming extension to the classical framework is proposed, to deal with decision variables, whose values can be set, and stochastic variables, which follow some probability distribution. However, these approaches deal with discrete domains and, although the ideas can be, in some cases, similar to the ones proposed here, both the techniques and the modeled problems are necessarily different, since we consider continuous domains.

6 Conclusions and Future Work

This paper presents a framework for probabilistic constraint reasoning, in which constraint reasoning is extended to obtain *a posteriori* conditional probability distributions from *a priori* distributions and evidence that is collected (possibly subject also to some degree of uncertainty). A number of properties of the framework are discussed and proven, and algorithms to implement it are presented. The framework has already been successfully applied to inverse problems [5], and the authors are now considering other types of engineering and biomedical applications, not only to further test the framework but also to adapt the algorithms presented in order to improve performance.

Another direction of future work is to address design problems. In this case, some of the variables are controlled by the user (decision variables) whereas others are outside user control (*state of nature* variables). Again the probabilistic approach will extend the standard safe approach of constraint reasoning. Safe reasoning aims at finding values for the system parameters, within some initial intervals, which satisfy the system constraints for *all* possible values of the *state of nature* variables, thus requiring some form of (universally) quantified constraints. Probabilistic reasoning will instead assume probability distributions for *state of nature* variables and will aim at obtaining values for the system parameters that guarantee the system to work properly with some degree of probability (e.g. for more than 95% of the time).

Acknowledgements This work was supported by Project PRECISE (POSI/EIA/59786/2004), funded by Portuguese Foundation for Science and Technology.

References

[1] J. Bellone, A. Chamard, and A. Fischler. Constraint logic programming decision support systems for planning and scheduling aircraft manufacturing at dassault aviation. In *Proc. of the 3rd ICPAP*, pages 63–67, Paris, 1995.

[2] F. Benhamou, D. McAllester, and P. van Hentenryck. CLP(intervals) revisited. In *ISLP*, pages 124–138. MIT Press, 1994.

[3] S. Bistarelli, U. Montanari, F. Rossi, T. Schiex, G. Verfaillie, and H. Fargier. Semiring-based csps and valued csps: Frameworks, properties,and comparison. *Constraints*, 4(3):199–240, 1999.

[4] E. Carvalho, J. Cruz, and P. Barahona. Probabilistic reasoning with continuous constraints. *AIP Conference Proceedings*, 936(1):105–108, 2007.

[5] E. Carvalho, J. Cruz, and P. Barahona. Probabilistic reasoning for inverse problems. In *Advances in Soft Computing*, volume 46, pages 115–128. Springer, 2008.

[6] H. Collavizza, F. Delobel, and M. Rueher. A note on partial consistencies over continuous domains. *Lecture Notes in Computer Science*, 1520:147–161, 1998.

[7] J. Cruz and P. Barahona. Constraint reasoning in deep biomedical models. *Journal of Artificial Intelligence in Medicine*, 34:77–88, 2005.

[8] H. Fargier and J. Lang. Uncertainty in constraint satisfaction problems: a probabilistic approach. In *Proc. of ECSQARU*, pages 97–104. Springer-Verlag LNCS 747, 1993.

[9] J. Haigh. *Probability Models*. Springer, 2002.

[10] J. Halpern. *Reasoning about Uncertainty*. MIT Press, 2003.

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

[12] A. N. Kolmogorov. *Foundations of the Theory of Probability*. Chelsea Publishing Company, 1950.

[13] V. Kreinovich. Probabilities, intervals, what next? optimization problems related to extension of interval computations to situations with partial information about probabilities. 29:265–280, 2004.

[14] J. Lamperti. *Probability. A Survey of the Mathematical Theory*. Wiley, 1996.

[15] O. Lhomme. Consistency techniques for numeric CSPs. In *Proc. of the 13th IJCAI*, pages 232–238, 1993.

[16] R. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, 1966.

[17] W. Older and A. Vellino. Constraint arithmetic on real intervals. pages 175–195, 1993.

[18] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

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

[20] N. Shazeer, M. Littman, and G. Keim. Constraint satisfaction with probabilistic preferences on variable values. In *Proc. of National Conf. on AI*, 1999.

[21] T. Walsh. Stochastic constraint programming. In *ECAI*, pages 111–115. IOS Press, 2002.