Global Consistency for Continuous Constraints [⋆]

D. Haroud and B. Faltings

Swiss Federal Institute of Technology, Ecublens 1015-Lausanne, Switzerland

Abstract. This paper provides a technique for solving general constraint satisfaction problems (CSPs) with continuous variables. Constraints are represented by a hierarchical binary decomposition of the space of feasible values. We propose algorithms for path- and higher degrees of consistency based on logical operations defined on this representation and demonstrate that the algorithms terminate in polynomial time. We show that, in analogy to convex temporal problems and discrete row-convex problems, convexity properties of the solution spaces can be exploited to compute minimal and decomposable networks using path consistency algorithms. Based on these properties, we also show that a certain class of non binary CSPs can be solved using strong 5-consistency.

1 Introduction

In the general case, constraint satisfaction problems (CSPs) are NP-complete. Trying to solve them by search algorithms, even if theoretically feasible, often results in prohibitive computational cost. One approach to overcome this complexity consists of pre-processing the initial problem using *propagation algorithms*. These algorithms establish various degrees of local consistency which narrow the initial feasible domain of the variables, thus reducing the subsequent search effort. Traditional consistency techniques and propagation algorithms — such as the Waltz propagation algorithm— provide relatively poor results when applied to continuous CSPs: they ensure neither completeness nor convergence in the general case (a good insight into the problems encountered can be found in [1]). However, Faltings [5] has shown that some undesirable features of propagation algorithms with interval labels must be attributed to the inadequacy of the propagation rule and to a lack of precision in the solution space description. He has also demonstrated that the problem with local propagation could be resolved by using *total constraints* on pairs of variables. Lhomme [10] has identified similar problems and proposed an interval propagation formalism based on bound propagation.

Van Beek's work on temporal reasoning [14] using Helly's theorem has shown the importance of path-consistency for achieving globally consistent labellings. In certain cases, path-consistency algorithms are difficult to implement in continuous domains because they require intersection and composition operations on constraints. We propose a constraint representation by recursive decomposition which allows implementation of these operations. This allows us to apply Helly's theorem to general continuous constraint satisfaction problems. The results obtained for temporal CSPs could therefore be extended to more general classes of continuous CSPs.

[⋆] Published in: "Lecture notes in computer Science 874: Principles and Practice of Constraint Programming", Alan Borning (ed.), Springer Verlag,1994

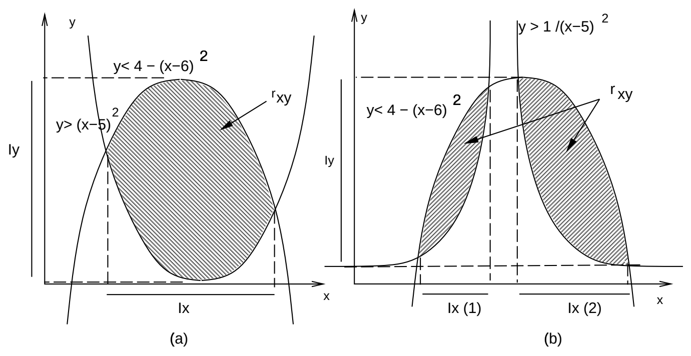

Fig. 1. *Figure (a) illustrates a binary relation R_{xy} given intensionally by the two inequalities $y > (x-5)^2$ and $y < 4-(x-6)^2$. R_{xy} determines the region r_{xy} and is both y- and x-convex: the projection of r_{xy} respectively over the x and y axes yields single bounded intervals (resp. I_x and I_y). In Figure (b), the relation R_{xy} is given intensionally by the constraints $y > 1/(x-5)^2$ and $y < 4-(x-6)^2$. In this last case, the relation is only y-convex since its projection over the x-axis yields two distinct intervals I_{x1} and I_{x2}.*

In the following, a continuous CSP (CCSP),$(P = (V, D, R))$, is defined as a set V of *variables* $x_1, x_2, \ldots x_n$ taking their values respectively from a set D of continuous *domains* $D_1, D_2, \ldots, D_n$ and constrained by a set of *relations* $R_1, \ldots, R_m$. A *domain* is an interval of $\mathcal{R}$. A *relation* is defined intensionally by a set of algebraic equalities and inequalities (see figure 1). A relation R_{ij} is a total constraint: it takes into account the whole set of algebraic constraints involving the variables i and j. Each variable has a *label* defining the set of possible consistent values. The label L_x of a variable x is represented as a set of intervals $\{I_{x,1} = [x_{min,1} \ldots x_{max,1}], \ldots\}$.

2 Constraint and Label Representation

Constraints on continuous variables are most naturally represented by algebraic or transcendental equations and inequalities. However, as Faltings [5] has shown, this leads to incomplete local propagation when there are several simultaneous constraints between the same variables. More importantly, making a network path-consistent requires computing the intersection and union of constraints, operations which cannot be performed on (in)equalities. It is therefore necessary to represent and manipulate the sets of feasible value combinations explicitly.

Providing each variable with an interval label implicitly represents feasible regions by enclosing rectangles or hypercubes. As shown in Figure 2, this is not powerful enough for region intersection operations. To define a more precise and yet efficient representation, we observe that most applications satisfy the following two assumptions:

- each variable takes its values in a bounded domain (bounded interval)
- there often exists a maximum precision with which results can be used.

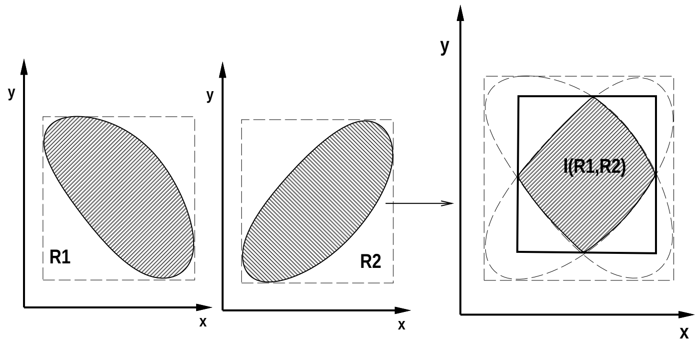

Fig. 2. *The enclosing rectangle of an intersection of regions R_1 and R_2 is in general different from the intersection of the enclosing rectangles of R_1 and R_2.*

Provided that these two assumptions are verified, a relation $R_{x_1 \ldots x_k}$ can be approximated by carrying out a hierarchical binary decomposition of its solution space into 2^k-trees (quadtrees for binary relations,octrees for ternary ones etc. . .)(see Figure 3). A similar representation has recently been proposed by Tanimoto for representing spatial constraints [13]. When a relation is determined by *inequalities*, it can be approximated by a 2^k-tree where each node represents a k-dimensional cubic sub-region of the original domain (i.e. the domain over which the decomposition is carried out). A node has one of three possible states:

– *white*: if the region it defines is completely legal
– *gray*: if the region is partially legal and partially illegal
– *black*: if the region is completely illegal

When a black or white node is identified, the recursive division stops. Each gray k-dimensional cube is decomposed into 2^k smaller ones whose sides are half as long. Unless the boundaries of a region are parallel to the coordinates axes, infinitely many levels of representation would be required to precisely represent a region. However, since the minimum granularity is fixed, any gray node with a smaller size than the minimum granularity can be declared black and the decomposition stops.

Equalities In the case of equality constraints, a strict application of the binary decomposition into a 2^k-tree as described would amount to pursuing the decomposition to infinity since an infinite degree of precision is required to represent single point solutions. We can avoid this problem by exploiting the fact that many practical applications require a limited degree of precision and it is thus admissible to treat equalities with a certain error range. Presently, our system translates strict equalities $f(x_1, \ldots x_k) = C$ into a weaker form, $f(x_1, \ldots x_k) = C \pm \varepsilon/2$, where ε is the fixed maximum precision, as defined for inequalities. This amounts to replacing each equality by two inequalities.

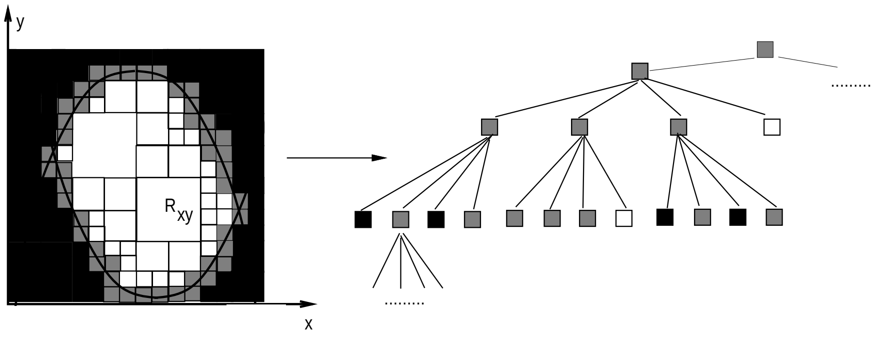

Fig. 3. *A binary relation R_{xy} can be approximated by carrying out a hierarchical binary decomposition of its solution space into a quadtree*

3 Consistency Algorithms Using 2^k-Trees

Path consistency algorithms, such as `PC-1` [12] and `PC-2` [11] require the application of the following update rule defined on constraints:

$$C'_{ij} = C_{ij} \oplus \prod^{k} (C_{ik} \otimes C_{kj}) \quad (1)$$

This relaxation operation uses two binary operators (intersection and composition, denoted respectively by $\oplus$ and $\otimes$) and a unary one (projection, denoted by $\prod$), which can be defined on 2^k-trees. Since all variables are decomposed within the same interval, *intersection* is simply the logical intersection of the corresponding quadtrees and can be carried out efficiently. *Composition* can be implemented by first extending the k-dimensional constraints into $k+1$-dimensional space and then projecting back the result into k-dimension, as shown in Figure 4. Given an ordering $white < gray < black$, rules for determining the feasibility of a node obtained by one of these operators can be expressed as follows:

i. $color(node_1 \oplus node_2) = Max(color(node_1), color(node_2))$
ii. $color(node_1 \otimes node_2) = Max(color(node_1), color(node_2))$
iii. $color(\prod^{x}(node_1)) = Min(color(node_i))$
where $node_i$ are the nodes having $node_1$ as facet.

The operators required for path consistency algorithms (and their generalization for higher degrees of consistency) can therefore be implemented as straightforward logical rather than numerical operations.

At each relaxation step described by eq. 1 using operations on 2^k-trees, the intervals contained in the involved labels are constructed by an implicit binary search: each successive relaxation step refines the interval bounds to an interval half the size of the previous one until maximum granularity is reached. Consequently, the decomposition into 2^k-trees has the important advantage of ruling out infinite cycling of the propagation algorithm as observed for the Waltz algorithm applied to continuous domains. While the Waltz algorithm performs slow and unstable fixed point iterations, the binary search method using the 2^k-tree decomposition guarantees stability and convergence.

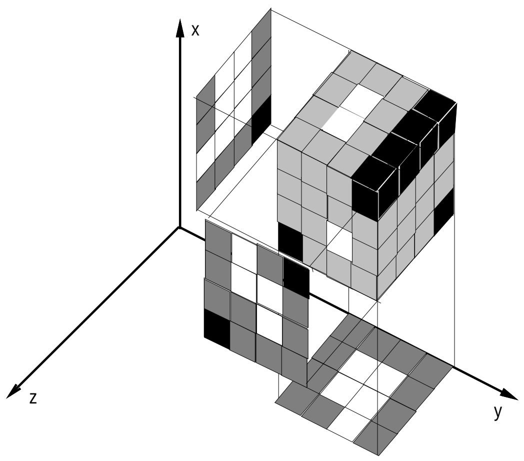

Fig. 4. *Information on a 3-dimensional node can be simply derived by composing its facets (2-dimensional nodes), and vice versa, information on a 2-dimensional node can be obtained by projecting the 3-dimensional node over one of its facets*

N-ary CSPs In many realistic problems, the constraints are not binary, but n-ary. However, each n-ary constraint can be reduced to a set of ternary constraints without loss of information. An n-ary algebraic relation, $C(x_1, \ldots x_n)$, can be transformed into a set of ternary algebraic expressions by:

i. replacing iteratively in C each sub-expression $< x_i$ operator $x_j >$ by a new variable x_{n+1}
ii. adding a ternary equality constraint $x_{n+1} =< x_i$ operator $x_j >$

The process stops when C itself becomes ternary. This transformation is only based on symbolic manipulations and consequently, no information is lost in the solution space description. For example, the 5-ary CSP with one constraint, $(x-y)^2 + \frac{(z+t)}{u} > 2$, can be translated into a ternary one with three constraints: $w_1^2 + (w_2/u) > 2$, $w_1 = x - y$, $w_2 = z + t$. Hence, addressing n-ary continuous CSPs amounts to giving the ternary counterparts of the algorithms and representation used for solving binary continuous CSPs.

Constructing 2^k-tree representations A total binary constraint R_{xy} is given intensionally by a set of algebraic equations $(C_1 \ldots C_l)$. The quadtree approximation T_{xy} of a binary relation R_{xy} can be obtained as follows:

```
For each $C_i \in (C_1 \ldots C_l)$ Do
  1. build a quadtree representation $T_{xy}^i$ for the basic constraint $C_i$
  2. $T_{xy} = T_{xy} \oplus T_{xy}^i$
```

Constructing the quadtree representation of an individual algebraic constraint, requires a procedure for determining the color of each sub-region (rectangle) created by the recursive decomposition. Two cases have to be considered:

i. If the constraint curve determines a *transverse segment* within the considered rectangle, testing for the rectangle color amounts to finding an intersection of its boundaries with the curve (see Figure 5). This test requires iterative numerical analysis in the general case.
ii. If not (i.e. the curve is *closed* within the considered rectangle), a pre-processing phase must be carried out in order to split the curve into transverse segments. This can be done for example by carrying out a binary search for determining a division which intersects the constraint curve. (see Figure 5)).

Computing octrees for ternary relations can be carried out in a similar manner.

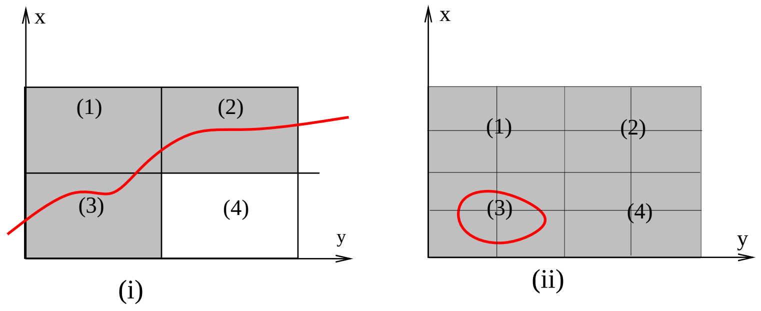

Fig. 5. *In case (i), the constraint curve determines a transverse segment: if the boundaries of a rectangle intersect the curve , the rectangle is gray (rectangle(1)). If not it suffices to test if one vertex of the rectangle satisfies the constraint to know if it is white or black (rectangle (4)). In case ii, a rectangle might be gray even if all of its vertices satisfy (or do not satisfy) the constraint (rectangle(3)). A preliminary search must be carried out to determine the divisions where the constraint curve is transverse.*

4 Global Consistency in Constraint Networks

In a *minimal* network, all the constraints are as explicit as possible and any value of the labels can be extended to a solution. In a *decomposable* network, the search for a solution is *backtrack-free* (the search process can generally be carried out in linear time). We show that certain convexity properties of the solution space make it possible to compute minimal and decomposable networks in polynomial time for continuous CSPs.

Encouraging results have been obtained for continuous CSPs in the domain of temporal reasoning: Dechter, Meiri and Pearl [2] have shown that for simple temporal problems (STP), where labels have to be convex intervals (i.e: disjunctive constraints are not allowed), the minimal constraint network can be constructed in polynomial time by ensuring path consistency. Similar results have been obtained by Van Beek [14] on a subset of the Allen's interval algebra excluding the binary relation $\neq$. Recently,Van Beek [15] has generalized the convexity property to the case of discrete CSPs: *row-convexity* guarantees the minimality and the decomposability of a path-consistent constraint network.

Although the convexity properties exploited in temporal and row-convex discrete problems derive mainly from results in the continuous domain (see Helly's theorem for convex sets [14]), no framework has been defined to exploit them in the case of general continuous CSPs. This is because the restriction imposed by the convexity condition on algebraic continuous solution spaces is too strong. In this work, we show that the *axis-convexity* property (a weaker condition) is sufficient for generalizing the results obtained in simple temporal [2] and row-convex discrete [15] domains to continuous CSPs.

In simple temporal problems (STPs) constraints take the form of bounded differences $b_1 < x_i - x_j < b_2$ where $[b_1 b_2]$ has to be a *single* interval. This condition amounts to saying that each variable takes its value within a single interval (convex interval). Path consistent STPs can be solved by backtrack-free search. The key observation is that this solution requires the convexity property *only* for each individual variable domain. Hence, generalizing to non-temporal continuous CSPs would amount to imposing convexity conditions only on the *projections* of the solution space over the different axes involved (the convexity condition is required only on projected intervals).

Consequently, to generalize the results obtained for STPs, it is sufficient that the solution space verifies partial convexity properties (convex projections). The weaker form of partial convexity that can be used is arcwise connectivity. A region R is arcwise connected if for any pair of points x and y of R, there exists a path connecting x and y which is entirely within R. An arcwise connected region yields convex outer projections over the axes involved. Convex outer projections aid in determining a tighter consistent approximation of the solution space but are not sufficient to guarantee that the resulting bounds are minimal. This is due to the fact that even if the outer projection is convex, this property is not necessarily preserved for a subprojection of the solution space. For this reason, we define a new category of partial convexity called *axis-convexity*. This property is more restrictive than arcwise connectivity but guarantees the convexity of any subprojection. We show in the next sections how this property can be used to determine minimal approximations of the solution space in polynomial time complexity.

Definition 1 ***: Axis-Convex Region***
Let r be a region defined by a set of algebraic or transcendental constraints on n variables $x_1 \ldots x_n$. r is said to be x_k-convex in the domain D_{x_k} if, for any two points q_1 and q_2 in r such that the segment $q_1 q_2$ is orthogonal to x_k, $q_1 q_2$ is entirely contained in r.

The *axis-convexity* requirement is clearly weaker than convexity: a k-ary relation, defined on a set of k variables $V = x_1, \ldots, x_k$ and determining a convex region has convex projections for each variable x_i of V for any value of x_i. However, the converse is not true, a region may have convex projections for each involved variable x without being convex.

4.1 Convex binary CCSPs

Let us first describe how convexity properties can be exploited in the case of binary constraints. The case of n-ary constraints will be dealt with later on. We define:

Definition 2 *: **Axis-Convex Relation***
A binary relation R_{x_i,x_j} is x_k-convex (where $k \in \{i, j\}$) if it determines a x_k-convex region in the domain D_{x_k}.

Definition 3 *: **x-Intersection***
The x_i-intersection of two bi-dimensional regions, $r^1_{x_i,x_u}$ and $r^2_{x_i,x_v}$, is the intersection of their projection over the x_i axis.

Definition 4 *: **Convex constraint network***
A constraint network representing a CCSP (V,D,R) is convex if for all relation $R_{x_i,x_j} in R$, Rx_i, x_j is x_k-convex for each k in $\{i, j\}$.

Continuous constraint satisfaction problems (CCSPs) having convex constraint network representations are the generalized counterparts of simple temporal problems (STP) as defined in [2]. Note finally that CCSPs including disjunctive or non-linear constraints may admit no convex constraint network representation since these types of constraints often create splits in the solution space.

Now we are in position to extend the theorems of Van Beek (theorem 1 and 4 of [15]) to the case of CCSPs. We first have to extend the lemma on which his proofs are based. This can be done as follows:

Lemma 1 *Let F be a finite collection of x-convex regions in R^2. If F is such that every pair of regions has a non null x-intersection, then the x-intersection of all these regions is not null (i.e: there exists at least one value v for x so that each region $r_{x,y}$ contains a point (v, y_i), where y_i is a possible value for y)*

Proof. This lemma is a direct application of Helly's theorem to the case of R^2.
We can generalize the results given in [15] as follows:

Theorem 1 *Let N be a path consistent binary constraint network. If the network is convex, it is also minimal and decomposable. If it is not convex, a consistent instantiation can be found without backtracking if there exists an ordering of the variables $x_1, \ldots x_n$ such that each relation of N R_{x_i,x_j}, $1 \leq j \leq i$, is x_i-convex.*

Proof. Analogous to the ones given in [15] (and based on the generalization of the backtrack-free instantiation algorithm proposed in this work).

4.2 Convex n-ary CCSPs

As stated before, generalizing to n-ary CCSPs the results described before for binary CCSPs amounts to giving the ternary counterpart of theorem 1.

Global consistency for ternary CCSPs The x-convexity property generalizes straightforwardly to the case of non binary CCSPs. In the case of ternary constraints, the generalization of lemma 1 can be used to prove the decomposability of the constraint network only if each pair of ternary relations has a non null x-intersection. Two ternary relations $R_{i_1,j_1,k}$ and $R_{i_2,j_2,k}$ have a non null k-intersection when each subset of *five* variables (i_1, i_2, j_1, j_2, k) are consistently labelled. In the particular case where each pair of ternary constraints has two variables in common,(i.e: $i_1 = i_2$ or j_1 or j_2), the number of variables that must be consistently labelled reduces to four and strong 4-consistency guarantees that the network is decomposable. Hence, theorem 1 generalizes to ternary constraints as follows:

Theorem 2 *A ternary constraint network which is convex and strongly 5-consistent is minimal and decomposable. Furthermore, in the particular case where each pair of relations share two variables, strong 4-consistency is enough to ensure that a convex ternary constraint network is minimal and decomposable.*

Since the translation of an n-ary network into a ternary one is done at the cost of increasing the number of variables, the practicality of 5-consistency for n-ary CCSPs is still an open question. This result is mainly intended to provide a theoretical bound for solving certain classes of n-ary CCSPs in a complexity better than exponential.

4.3 Non-convex CCSPs

A general CCSP may admit no convex constraint network representation. Moreover, even if the initial problem is convex, consistency algorithms may not preserve this property since intersecting two non convex — even if axis-convex— regions may result in an arbitrary number of distinct sub-regions. We can distinguish three classes of CCSPs:

i. CCSPs where all the relations determine convex regions
ii. CCSP where each relation determining a non arcwise connected solution space is constituted by a set of convex regions.
iii. CCSPs where there exist non convex regions

In case *i*, since the intersection of two convex regions is necessarily convex (and hence axis-convex), consistency algorithms will preserve the convexity of the constraint network representation. Hence, problems of this first category can be solved, with no further search, using partial consistency algorithms (as stated in theorems 1 and 2). In case *ii*, the problem can be decomposed into convex sub-problems (one for each possible combination of convex sub-region), for which each sub-problem is of type *i*. A solution to the whole problem can be determined by solving each sub-problem individually and then combining their solutions. Even if the complexity is, in this case, exponential in the number of disjoint convex sub-regions, the computational effort can be bounded *a priori* since consistency algorithms cannot create new case splits in the individual sub-problems. In the last case, the splitting problem (similar to the one described in [9]) may occur and the complexity is difficult to estimate. In the best case, the consistency algorithm may create a convex constraint network from a set of non-convex relations. In the worst case however, the intersection of each pair of non convex regions may result in an unbounded number of disjoint new sub-regions which can in turn split again. Practical solutions (such as stopping the splitting process when the maximum precision is reached) can be used to bound the combinatorial explosion, but in general the complexity remains exponential for CCSPs of type *iii*.

5 Complexity of Consistency Algorithms

The complexity of the intersection, composition and projection operators on 2^k-trees can be roughly estimated in terms of the number of nodes generated by each operation. $O(2^{k*s/\varepsilon})$ (where s is the maximum domain size and ε the tightest interval size accepted for variables) gives a rough approximation of the complexity. This measure assumes that, in the worst case, a 2^k-tree resulting from a given operation is complete. A more realistic measure can be done in terms of the number of gray nodes generated, since the recursive quartering stops as soon as a node color is set to white or black. We can show that this measure is a function of the boundary size of the solution space. Furthermore, 2^k-tree structures are by nature well-adapted to parallel processing. Parallel implementation of the intersection, composition and projection are likely to be very efficient.

Convex Binary CCSPs The algorithm PC-2 can be implemented using eq. 1 by way of the `revise` function. According to the definitions of $\oplus$ and $\otimes$ for 2^k-trees, the relaxation operation described by eq. 1 is monotonic. Moreover, since the region decomposition into 2^k-trees discretizes the solution space, the fact that `PC-1` (and hence `PC-2`) terminates and computes a path-consistent network using the relaxation operation $C'_{ij} = C_{ij} \oplus \prod^k (C_{ik} \otimes C_{kj})$ can be shown in a manner similar to the case for discrete-domain CSPs (see [12]). The worst case running time of `PC-2` occurs when each revision step suppresses only one node from the considered relation (i.e. the node becomes black), hence:

Theorem 3 *PC-2 computes the path consistent network representation of binary CCSPs, (V, D, R), in $O(2^{(2*s/\varepsilon)}n^3)$ where s is the largest interval size in D and ε the tightest interval size accepted for variables of V.*

According to theorem 1, when the path consistent network computed by PC-2 is convex, it is also minimal and decomposable. Similarly, we can demonstrate that strong 5-consistency can be ensured for a ternary CCSP in $O(2^{(3*s/\varepsilon)}n^5)$.

Non convex CCSPs During the construction and propagation of 2^k-trees, the case in which a single region is split into several can be reliably detected. At this point the algorithm branches and explores both regions separately (a new CCSP is generated). The pathological case where an infinite number of sub-regions are generated is avoided in practice, since regions smaller than the maximum precision are not explored. However, the worst case complexity is clearly exponential.

6 Conclusion

In this paper we present a generalization of the results obtained for convex temporal problems and discrete row-convex problems to more general classes of continuous CSPs (convex CCSPs). The main contributions is to show that *partial convexity* properties of continuous solutions spaces can be exploited to compute solutions to CCSPs in polynomial time. A recursive decomposition scheme is proposed that solves the problem of representing total constraints for path consistency algorithms. The 2^k-tree decomposition amounts to performing the stable binary-search method which guarantees convergence according to numerical analysis results. The cycling problems, generally posed by fixed point iteration methods (such as those observed by Davis for the Waltz algorithm [1]) are consequently avoided. Finally, we show that solving non-convex CCSPs remains inherently costly, but decomposition methods can be proposed and might be of practical interest for many particular applications.

7 Acknowledgments

We thank the Swiss National Science Foundation for sponsoring this research under contract No.5003-034269

References

1. Davis E. : *“Constraint propagation with interval labels”*, Artificial Intelligence 32 (1987)
2. Dechter R., Meiri I., Pearl J. : *“Temporal constraint networks”*, Artificial Intelligence 49(1-3) (1990)
3. Dechter R.: *“From local to global consistency”*, Proceedings of the 8th Canadian Conference on AI (1990)
4. Deville Y., Van Hetenryck P.: *“An efficient arc consistency algorithm for a class of CSP problems”*, Proceedings of the 12th International Joint Conference on AI (1991)
5. Faltings B.: *“Arc consistency for continuous variables”*, Artificial Intelligence 65 (2) (1994)
6. Freuder E.C.: *“Synthesizing constraint expressions”*, Comm. ACM 21 (1978)
7. Freuder E.C.: *“A sufficient condition for backtrack-free search”*, J. ACM 29 (1982)
8. Freuder E.C.: *“A sufficient condition for backtrack-bounded search”*, J. ACM 32 (1985)
9. Hyvönen E.: *“Constraint reasoning based on interval arithmetic: the tolerance propagation approach”*, Artificial Intelligence 58(1-3) (1992)
10. Lhomme O.: *“Consistency techniques for numeric CSPs”*, Proceedings of the 13th International Joint Conference on AI (1993)
11. Mackworth A.: *“Consistency in networks of relations”*, Artificial Intelligence 8 (1977)
12. Montanari U.: *“Networks of constraints: fundamental properties and applications to picture processing”*, Inform. Scie. 7 (1974)
13. Tanimoto T.: *“A constraint decomposition method for spatio-temporal configurations problems”*, Proceedings of the the 11th National Conference on AI (1993)
14. Van Beek P.: *“Approximation algorithms for temporal reasoning”*, Proceedings of the 11th International Joint Conference on AI (1989)
15. Van Beek P.: *“On the minimality and decomposability of constraint networks”*, Proceedings of the 10th National Conference on AI (1992)