Arc Consistency for Continuous Variables

Boi Faltings
Laboratoire d'Intelligence Artificielle
Swiss Federal Institute of Technology (EPFL)
IN-Ecublens
1015 Lausanne
Switzerland

January 12, 1998

Abstract

Davis [1] has investigated the properties of the Waltz propagation algorithm with interval labels in continuous domains. He shows that in most cases, the algorithm does not achieve arc consistency, and furthermore is subject to infinite iterations.

In this paper, I show that the main reason for Davis' negative results lies in the way he formulates the propagation rule for the Waltz algorithm. For *binary* constraints, I propose a different propagation rule and show that it guarantees arc consistency upon quiescence of the propagation. Generalizations to n-ary constraints are possible but involve more complex geometry.

Arc consistency guarantees a minimal network only when the constraint graph is a tree. I show that the new formulation of the propagation algorithm rules out the possibility of infinite iterations for all tree-structured constraint networks, and thus obtain a general and reliable algorithm for arc consistency in continous domains.

1 Introduction

Many problem-solving tasks can be formulated as *constraint satisfaction problems*(CSP), which I define as follows:

> Given a set of variables $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ with associated domains $D_1, D_2, ..., D_n$, find all sets of value assignments to the variables such that a set of constraints $\{c_1, c_2, ..., c_k\}$ is simultaneously satisfied.

Propagation algorithms compute a concise description of the set of potential solutions in an efficient manner by locally propagating the effects of constraints. Algorithms for *arc consistency* [5] reduce the domains of the variables by eliminating those values which are inconsistent with related constraints and the allowed values of the other variables. The reduced domains are represented by *labels*. Freuder [3] has pointed out that if the constraint network is a tree, a solution can be found without backtracking from an arc consistent labelling.

For discrete and finite domains, labels can be represented explicitly as sets of values and there are efficient polynomial-time algorithms [6] for computing arc consistent labellings. I consider in particular the Waltz propagation algorithm [8], also known as algorithm AC-1 [6]. In this algorithm, a *propagation rule* **REFINE**(x, y) is applied iteratively to revise the label of y such that it contains only values consistent with the constraint $c(x, y)$ and the label of x, and similarly with x and y reversed. The algorithm stops when the propagation rule has been applied to all constraints without causing any change to the labels.

The seminal paper of Davis [1] investigates the properties of Waltz propagation algorithms for continous domains where labels are *intervals* of values. He shows that in most cases, such algorithms are incomplete and furthermore subject to infinite iterations. Hyvönen [4] has shown methods for computing consistent interval labellings in networks of equality constraints. The methods could be applied to inequalities by introducing additional slack variables, e.g. $f(x, y) > 0$ can be transformed into $f(x, y) = z$ where $z \in [0, \infty]$.

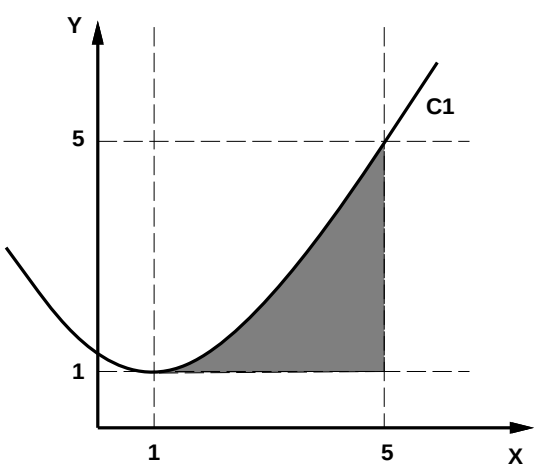
In this paper, I define a new Waltz propagation rule for inequality constraints and show that it results in a complete algorithm. I further show that cases where propagation leads to infinite iterations cannot occur when the constraint network is a tree. Thus, the algorithm is guaranteed to terminate in this only case where arc consistency guarantees a backtrack-free solution.

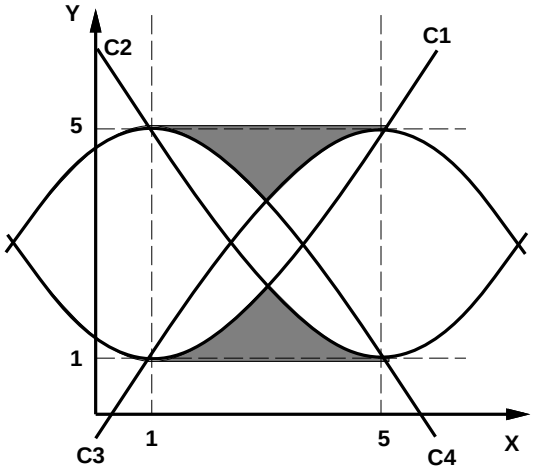
More precisely, I consider a set of variables $\{x_i\}$, restricted to finitely bounded domains, and a set of *binary* constraints $\{c(x_i, x_j)\}$ between them, called *individual constraints*. I assume that individual constraints are defined by inequalities of the form:

$$c(x, y) \geq 0$$

where $c(x, y) = 0$ is a continuous and differentiable curve in variables x and y. Some expressions might require restricting the domain of the variables to a subspace: for example, the expression $y - 1/x \geq 0$ is continuous only in the intervals $[-\infty..0^-]$ and $[0^+..+\infty]$. When initial intervals do not impose a sufficient restriction, the CSP must be solved for each subinterval separately. Equalities are represented by a pair of inequalities in both directions. The set of all individual constraints $c(x_i, x_j)$ between the same pair of variables defines a single *total constraint* $C^t(x_i, x_j)$. This total constraint is defined as one or more bounded *feasible regions* Q of value combinations which satisfy each of the individual constraints.

Each variable has a *label* which defines the set of possible consistent values. The label L_x of a variable x is represented as a finite set of continous and bounded intervals $\{I_x = [x_{min}..x_{max}], ...\}$. I define:





$$c : y \leq 0.25 * (x-1)^2 + 1$$

$$\begin{aligned} c1 : y &\leq 0.25 * (x-1)^2 + 1 \\ c2 : y &\leq 0.25 * (x-5)^2 + 1 \\ c3 : y &\leq 5 - 0.25 * (x-1)^2 \\ c4 : y &\leq 5 - 0.25 * (x-5)^2 \end{aligned}$$

Figure 1: *Propagation through the single constraint on the left does not lead to any modification of the labels* $x \in [1..5], y \in [1..5]$. *The same is true for each of the four constraints shown on the right. However,* $c1 \wedge c2$ *only allow the lower shaded region, and* $c3 \wedge c4$ *only the upper shaded region. Thus, there are no points which simultaneously satisfy* $c1 \wedge c2 \wedge c3 \wedge c4$. *This is not detected when constraints are propagated individually.*

Definition 1 *The label* L_x *of variable* x *is* arc consistent *with the label* L_y *of variable* y *if and only if:*

$$(\forall x_0 \in I_x \in L_x)(\exists y_0 \in I_y \in L_y)\,[(\forall c(x,y))c(x_0,y_0) \geq 0]$$

I assume that labels are always finite, but arbitrarily large intervals.

A labeling is called arc consistent *if the labels of all ordered pairs of variables are arc consistent, and* sound *if it contains all solutions to the constraint satisfaction problem. A Waltz propagation algorithm is* sound *if it computes a sound labelling, and* locally complete *if it computes an arc consistent labeling.*

Note that this definition only concerns arc consistency, not *global* consistency, which could not always be represented by a set of interval labels anyway. Only in tree-structured constraint networks is arc consistency equal to global consistency.

In Davis' version of the Waltz propagation algorithm [1], the problem of *early quiescence* occurs when the propagation rule results in no changes to the labelling in spite of the fact that it is not yet arc consistent. One reason for this is that Davis represents labels by *single* intervals which are often insufficient to represent an arc consistent labelling, as already observed for example by Hyvönen [4].

Another problem with Davis' propagation rule occurs when there are several constraints between the same pair of variables. For the single constraint shown on the left in Figure 1, the shown intervals $x \in [1..5], y \in [1..5]$ are arc consistent labels. Now consider the four simultaneous constraints between x and y shown on the right in Figure 1. For each constraint

REFINE(x, y)**:**

1. $I_y \leftarrow \{\}$
2. for all $I_x \in L_x$ do:
$$I_y \leftarrow I_y \cup \textbf{simple-propagate}(I_x, C^t(x, y))$$
3. $IU \leftarrow$ union of all intervals in I_y, where overlapping intervals are merged into single continous ones.
4. $L_y \leftarrow$ intersection of L_y with IU, computed as the union of overlapping subintervals in L_y and IU.

Figure 2: *Propagation rule for the Waltz algorithm.*

taken individually, the intervals $x \in [1..5]$ and $y \in [1..5]$ are arc consistent. However, taken together the four constraints do not allow any feasible solution. Davis' formulation of the propagation rule considers each constraint in isolation, is quiescent with the intervals shown and thus incomplete.

2 An improved propagation rule for binary constraints

Differently from the formulation used by Davis, I define:

Definition 2 *The* label L_x *of a variable* x *is a* set *of disjoint intervals* $L_x = \{I_1, I_2, ..., I_n\}$.

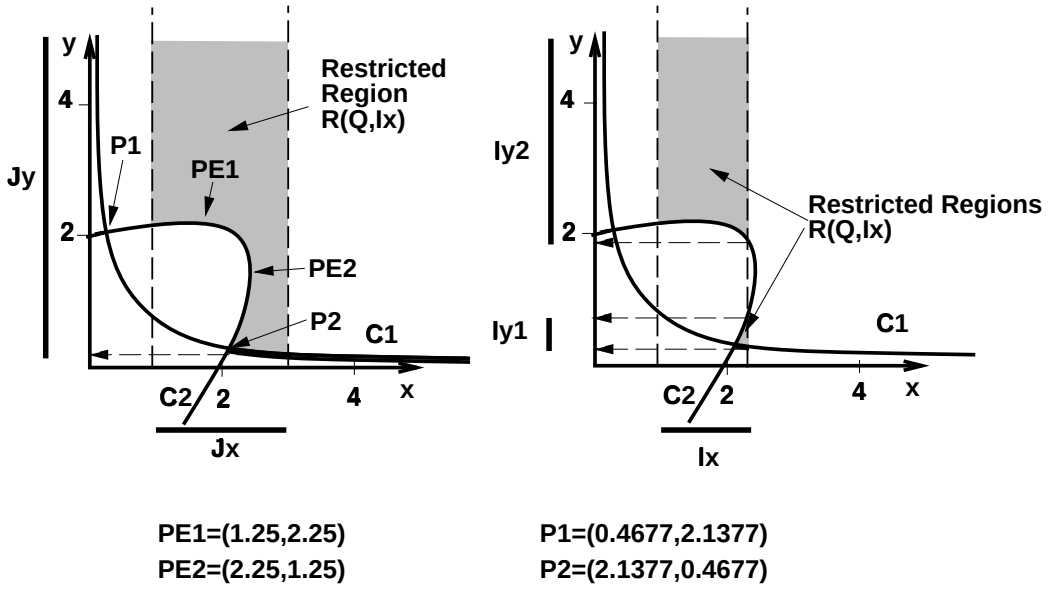
Definition 3 *The* total constraint $C^t(x, y)$ *is a set of bounded feasible regions* $C^t(x, y) = \{Q_1, Q_2, ..., Q_n\}$ *which contain exactly those combinations* (x, y) *which are consistent with all constraints* $c(x, y)$.

The propagation rule **REFINE**(x, y), shown in Figure 2, is responsible for propagating the label of a variable x through the total constraint $C^t(x, y)$ to the label of variable y. **REFINE** computes the new label L_y by intersecting the old label with the union of all values which are arc consistent with an interval in L_x. This latter computation is carried out by the function **simple-propagate** which is defined in the next section.

2.1 Propagation of a single interval

The function **simple-propagate**(I_x, Q) computes the intervals of y-values which occur in the subregions of a feasible region Q where $x \in I_x$. This computation is complicated by the fact that there may be several such subregions, and that their bounds in y are often given by local extrema of Qs boundary. Figure 3 illustrates an example of a propagation. I now present an algorithm which handles such cases by considering only local extrema of the regions with respect to the two coordinate axes, and intersections of the region boundaries among each other and with interval bounds. I define:

Definition 4 *For a feasible region* $Q \in C^t(x, y)$, *the set of* restricted regions $R(Q, I_x) = \{r_1, r_2, ...\}$ *are those connected subregions of* Q *whose* x*-coordinates are entirely contained within an interval* I_x.



$$C1: \quad y - 1/x \geq 0 (\text{restricted to } x, y \in [0^+..5])$$
$$C2: \quad (x-y)^2 + 2x + 2y - 8 \geq 0$$

Figure 3: *The constraint curves $C1$ and $C2$ between x and y define a region of feasible value combinations. Within a feasible region Q, only those value combinations whose x-coordinate falls into interval Ix are arc consistent with xs label. Propagation of the interval $J_x = [1..3]$ results in a single interval J_y for y, while propagating $I_x = [1..2.2]$ results in a pair of disjoint intervals I_{y1} and I_{y2}.*

Each restricted region $r \in R(Q, I_x)$ defines a single continous interval of arc consistent values for y, which is given as:

$$I_y = [min\{y|\exists(x,y) \in r\}..max\{y|\exists(x,y) \in r\}]$$

Thus, in the example of Figure 3, the set $R(Q, I_x)$ contains two regions r which define the intervals I_{y1} and I_{y2} for y containing the values which are arc consistent with $x \in I_x$.

I define the *local* extrema of a set S as follows:

Definition 5 *A* local maximum *in y of a set S is a set $I \subset S$: $\{(x, y_0)|x_1 \leq x \leq x_2\}$ such that there is a neighbourhood $N = [x_1 - \epsilon..x_2 + \epsilon]$, $\epsilon > 0$, where any (x', y') with $y' \geq y_0$ is in I. A* local minimum *is defined similarly, but with the inequality in y' reversed: $y' \leq y_0$. A* local extremum *is either a local maximum or minimum.*

Note that the extremum can be a point when $x_1 = x_2$, or a line segment when $x1 < x2$. As the regions are finitely bounded, the absolute minimum y of a region r is equal to the minimum of all *local* minima in y on the boundary of r, and likewise for the absolute maximum.

The set of local extrema on the boundaries of all restricted regions $r \in R(Q, I_x)$ is composed of two parts:

- intersections of Qs boundary with the interval bounds of I_x which can be local maxima or minima, and

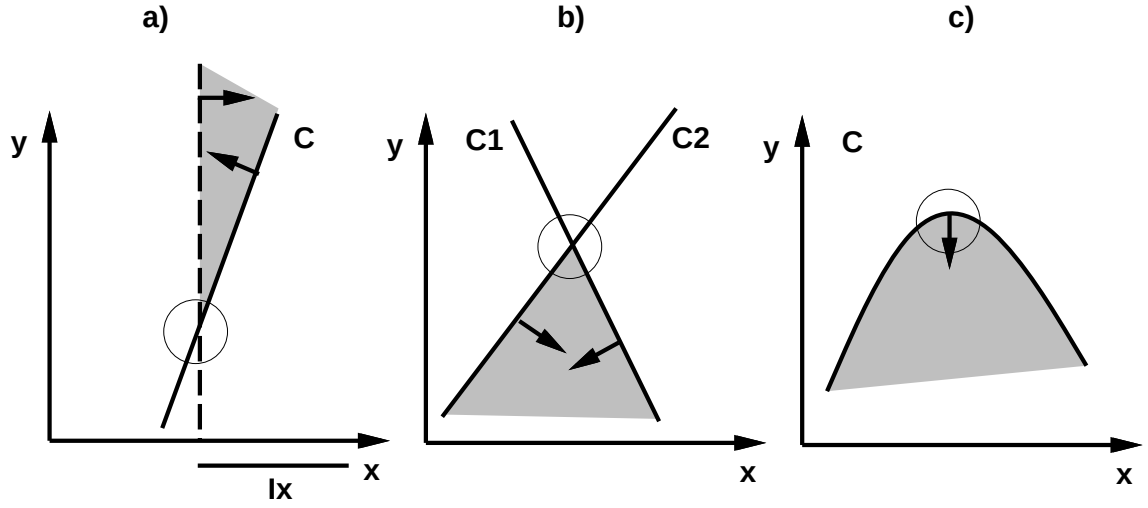


Figure 4: *Examples of local extrema of regions: a) local minimum at the intersection with an interval bound, b) local maximum at the intersection of two individual constraints, c) local maximum of an individual constraint.*

- local extrema of the boundary of Q whose x-coordinate falls within I_x - these can be local extrema of individual constraints or intersections of them.

Figure 4 shows examples of such extrema. A candidate set can be generated by computing all local extrema of constraints as well as all intersection points between constraints or constraints and interval bounds, and classified based on the gradients of the constraint curves at the point. Among the candidate set, only those points which satisfy all other individual constraints are actual extrema of a feasible region; all other points are eliminated based on this test. This computation requires numerical analysis procedures which depend on the type of constraints and are beyond the scope of this paper. What is important is that the complete set of all local minima and maxima of the restricted regions of all feasible regions of a total constraint $C^t(x, y)$ can be computed by local consideration of the individual constraints $c(x, y)$ only, without knowledge of the feasible regions themselves.

For the example of Figure 3, I have:

– no local extrema for $C1$.
– for $C2$, a local maximum in y at $PE1$ and in x at $PE2$.
– a local minimum in y at the intersection $P1$ of $C1$ and $C2$, and a local minimum in x at $P2$.

All of these points satisfy all individual constraints and are thus valid.

At each propagation step, only intersections with interval bounds must be computed. For the example on the right of Figure 3, I have the following intersections of the left bound:

– at (1,1) with $C1$, but this point does not satisfy $C2$ so it is invalid.
– at (1,2.2360) with $C2$, which is a local minimum in y.

For the right bound, I have the following 3 intersections:

– at (2.2, 0.4545) with $C1$, which is a minimum in y.
– at (2.2,1.6472) (minimum in y) and (2.2,0.7527) (maximum in y) with $C2$.

1. $I_y \leftarrow \{\}$.
2. compute the set MAX of local maxima and the set MIN of local minima of all restricted regions $r \in R(Q, I_x), Q \in C^t(x, y)$, and order the two sets of points according to their y-coordinate (these sets are obtained directly by consideration of individual constraints $c(x, y)$).
3. set $index \leftarrow 0$, consider the extrema e in MIN and MAX in decreasing order of their y-coordinate, where elements of MAX are always considered before elements of MIN, and do:
   (a) if $e \in MAX$, $index \leftarrow index + 1$. If $e \in MIN, index \leftarrow index - 1$.
   (b) if $index$ has just changed from 0 to 1, set $y_{upper} \leftarrow y - coordinate(e)$.
   (c) if $index$ has just changed from 1 to 0, set $y_{lower} \leftarrow y - coordinate(e)$ and add the interval $[y_{lower}..y_{upper}]$ to I_y.
4. **return** I_y.

Figure 5: *Algorithm* **simple-propagate***(I_x)*.

Furthermore, since y is bounded to $y \leq 5$, there is a local maximum at $y = 5$ for any interval in x.

The algorithm for **simple-propagate**(I_x), shown in Figure 5, is based on the idea that for a value y_0 to be arc consistent, it must only be known to fall within *some* region of $C^t(x, y)$ consistent with I_x, but it is not necessary to know which one. Executing the algorithm on the example of Figure 3, I group those extrema in y which fall within I_x into the sets:

$$\begin{aligned} MAX &= \{M_1 = (*, 5), M_2 = (1.25, 2.25), M_3 = (2.2, 0.7527)\} \\ MIN &= \{m_1 = (1, 2.360), m_2 = (2.2, 1.6472), m_3 = (2.2, 0.4545)\} \end{aligned}$$

where the "$*$" stands for an unknown x-coordinate within I_x.

Considering the extrema in decreasing order, I obtain the following trace:

Step	Extremum	$index$	y_{lower}	y_{upper}	I_y
0	none	0	-	-	{}
1	M_1	1	-	5	{}
2	M_2	2	-	5	{}
3	m_1	1	-	5	{}
4	m_2	0	1.6472	5	{[1.6472..5]}
5	M_3	1	-	0.7527	{[1.6472..5]}
6	m_3	0	0.4545	0.7527	{[1.6472..5], [0.4545..0.7527]}

which are in fact the intervals of y which are arc consistent with the $x \in I_x$.

I shall now prove that procedures **simple-propagate** and $REFINE$ actually guarantee a sound and locally complete Waltz propagation algorithm.

2.2 Soundness and completeness

I first define:

Definition 6 *The* index *of a set S is the difference between the number of local maxima and minima on its boundary $B(S)$: $i(S) = |MAX \in B(S)| - |MIN \in B(S)|$. For a y-coordinate y_0,* above(y_0, S) *is the index of the part of S whose y-coordinate is greater than y_0.*

and observe the following lemmas:

Lemma 1 *On a two-dimensional closed curve B, local minima and maxima occur in alternating order. The total number of local minima and maxima on B is equal, i.e. if B is the boundary of a region R, the index $i(R) = 0$.*

Proof: Consequence of the mean-value theorem and the continuity of B. The alternating circular order further implies an equal number of minima and maxima.

Lemma 2 *The index of every complete feasible region Q is $i(Q) = 0$, and for any y-coordinate y_0 for which there is no point $(x, y_0) \in Q$, $above(y_0, Q) = 0$.*

Proof: The index of the region is the sum of the indices of its boundaries, which are each equal to zero by Lemma 1. Given a y-coordinate y_0 with no point in Q, Q is either completely above y_0 or completely below y_0. If Q is completely above y_0, $above(y_0, Q) = i(Q) = 0$ and $below(y_0) = 0$ since there is no point of Q below y_0. An analogous argument holds for the case where Q is entirely below y_0. QED.

Lemma 3 *Consider a line $L : y = y_0$ intersecting a closed curve B, and the part I of B between two successive intersections with L. Either all points in I have a y-coordinate $\geq y_0$, and I has exactly one more local maximum than minimum in y, or all points in I have a y-coordinate $\leq y_0$, and I has exactly one more local minimum than maximum in y.*

Proof: Since I does not intersect L, it must lie either completely above or below L. If it lies completely above L, the two local extrema closest to the intersection points must both be maxima (Figure 6). By Lemma 1, extrema occur in alternating order, so that there must be exactly one more maximum than minimum within I. An analogous argument holds when I lies below L. QED.

This Lemma implies:

Lemma 4 *If there is a point (x, y_0) within a region r, $above(y_0, r) > 0$.*

Proof: Consider the line $L : y = y_0$ and a connected region r. If the point (x, y_0) lies within r and on L, L must intersect a boundary B of r. $above(y_0, r)$ is the sum of the indices of all segments I of all boundaries B which fall above L, of which there must be at least one. By Lemma 3, this index is always equal to one, and thus any sum of more than one such indices must be greater than zero. QED.

Lemmas 2 and 4 imply:

Lemma 5 *For any value y_0 and any region r, $above(y_0, r) \geq 0$.*

Proof: By the law of the excluded middle, either there exists a point $(x, y_0) \in r$ and $above(y_0, r) > 0$ by Lemma 4, or there does not exist such a point and $above(y_0, r) = 0$ by Lemma 2. QED.

This now allows us to show the following:

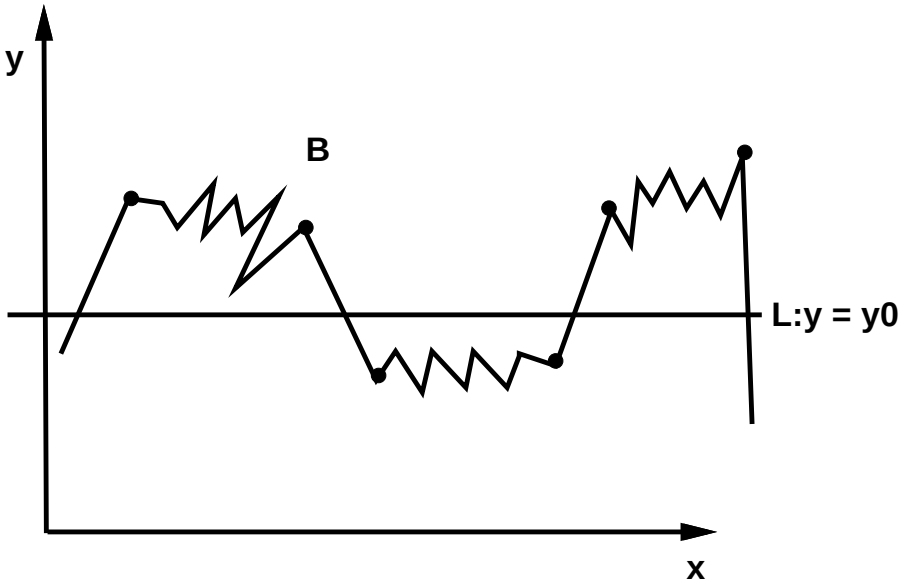


Figure 6: *Counting maxima and minima on a curve B intersecting L.*

Theorem 1 *For a y-coordinate y_0 and a feasible region Q, there exists a region $r \in R(Q, I_x)$ which contains a point (x_0, y_0) if and only if $\sum_{r \in R(Q,I_x)} above(y_0, r) > 0$. Furthermore, there exists a point $(x_0, y_0) \in C^t(x, y)$ with $x_0 \in I_x$ if and only if $\sum_{Q \in C^t(x,y)} \sum_{r \in R(Q,I_x)} above(y_0, r) > 0$*

Proof: Assume that there was no region r containing a point (x, y_0). Then for all regions r, $above(y_0, r) = 0$ by Lemma 2 and thus their sum cannot be greater than 0. Conversely, if there is a region r containing y_0, by Lemma 4 $above(y_0, r) > 0$. Since by Lemma 5, $above(y_0, r) \geq 0$ for any region r, the sum must be positive. The same argument extends to all restricted regions r of all feasible regions $Q \in C^t(x, y)$. QED.

I now return to the propagation algorithm and prove:

Theorem 2 *The label L_y returned by the propagation rule* **REFINE**(x, y) *is always arc consistent with L_x.*

Proof: Note that in procedure **simple-propagate**, the variable *index* always denotes the value of $\sum_{Q \in C^t(x,y)} \sum_{r \in R(Q,I_x)} above(y_0, r)$, where y_0 is the y-coordinate of the current point e, $Q \in C^t(x, y)$ and $I_x \in L_x$. By Definition 4 and Theorem 1, any value of y for which *index* is positive is arc consistent with L_x. The values returned for L_y are a subset of those for which $index > 0$, and thus L_y is arc consistent with L_x. QED.

This theorem implies that:

Theorem 3 *A Waltz propagation algorithm using the propagation rule defined above is* locally complete.

Proof: I assume that the propagation has reached quiescence, i.e. for any pair of variables x and y, the propagation has not resulted in any changes to the labels. Thus, by Theorem 2, for any pair of variables x and y the label L_x is arc consistent with L_y, and vice versa. Thus, the labelling is arc consistent and the algorithm locally complete. QED.

Conversely, I can also show that:

Theorem 4 *The propagation rule defined above never creates a label which is not sound. Consequently, a Waltz propagation algorithm using it is also* sound *whenever it starts with a sound labelling, i.e. it never eliminates any solutions.*

Proof: I use an inductive proof. The base of the induction is that the initial labels of any pair of variables x and y are sound. **REFINE**(x, y) will eliminate a value y_0 from L_y exactly when there is no interval $I_x \in L_x$ and no subregion $r \in R(Q, I_x), Q \in C^t(x, y)$ such that $above(y_0, r) > 0$. By Definition 4 and Theorem 1, this means that there is no point $(x_0, y_0) \in Q$ such that $Q \in C^t(x, y), x_0 \in L_x$. Consequently, y_0 cannot be consistent with L_x and $C^t(x, y)$, and not part of any solution to the CSP. Thus the propagation algorithm is sound. QED.

3 Infinite Iterations

The second problem with the Waltz algorithm in continuous domains identified by Davis is what he terms *looping*: the algorithm never reaches quiescence. Davis gives the example of the constraints $x = y$ and $x = 2y$, where the propagation algorithm will take an infinite number of steps to find the solution $x = y = 0$ from initial intervals $x \in [0..1]$ and $y \in [0..1]$. The same problem can also occur with inequalities. Even more problematic is the fact that in certain cases, each propagation step may create new subintervals, resulting in an unbounded explosion of label size and thus computational complexity (see [4] for a discussion of this problem).

The looping problem is not surprising, since the computation steps carried out by the propagation algorithm in many cases turns out to implement a *fixed-point iteration* whose convergence is slow and for which no reliable termination criterion can be given. In fact, it is possible to solve any system of nonlinear equations by posing it as a problem of satisfying constraints representing the equations. It would be very surprising if a propagation algorithm could provide a simple and reliable solution to this problem!

Arc consistency guarantees backtrack-free value assignment only when the constraint network is a tree [3], and is only of limited use in other cases. I now show that looping behavior can only occur when the constraint network contains cycles. This result shows that for the practically most useful cases, the algorithm obtained with the new formulation of the propagation rule is guaranteed to terminate in a finite number of iterations.

Causes of looping Theorem B.21 in [1] is valid for any constraint network, including one containing total constraints, and shows that that looping only occurs in the case of *self-dependencies*:

Theorem 5 *(Davis) Given a set of refinements, $S = \{R_1, ..., R_m\}$ on n variables, and a labelling L, then either there exists a series of refinements in S containing a self-dependency on L, or there exists a series of refinements which is not redundant which brings L to quiescence.*

Consequently, the propagation algorithm can exhibit the looping behavior only in the presence of self-dependent series of refinements.

Tree-structured constraint networks It is straightforward to show that when there is at most one total constraint between any pair of variables, self-dependencies occur only in the presence of cycles in the constraint network. However, I shall now show an even stronger result, namely the existance of a series of refinements which achieves an arc consistent labeling and whose length is linear in the number of variables:

Theorem 6 *When the constraint network is a tree, the following sequence of refinements achieves an arc consistent labelling:*

1. *Apply* **REFINE** *from the leaf nodes to their parents, and recursively up to the root of the tree.*

2. *Apply* **REFINE** *from the root node recursively back to the leaves.*

Proof: Consider a variable x with ancestor y in the tree. The algorithm applies **REFINE** between x and y exactly two times:

- **REFINE(x,y)** makes the label of y arc consistent with the label of x by Theorem 2. Subsequent modifications to y's label only eliminate values and cannot create values which are not arc consistent with x's label. Thus y's label will remain arc consistent with that of x.

- **REFINE(y,x)** makes the label of x arc consistent with the label of y by Theorem 2. Because the propagation rule is sound by Theorem 4, this operation cannot eliminate any values of x which are required to maintain y's label arc consistent with that of x; it thus remains arc consistent. Since there are no further modifications to the labels of x and y, they remain arc consistent when the algorithm terminates.

QED.

Since a tree of n nodes has $n-1$ arcs, this algorithm terminates with an arc consistent network in $2(n-1)$ applications of the **REFINE** operator.

4 Discussion

The main contribution of this paper is to disprove the pessimistic results of Davis [1] which imply that constraint propagation algorithms in continous domains are almost always incomplete even for tree-structured networks. I have shown that this negative result is due to the fact that all constraints between variables are considered in isolation. By using a network of *total constraints*, sound and locally complete propagation algorithms are indeed possible. In hindsight, this should not be surprising, as Mackworth [5] and Montanari [7] had already pointed out the generality of local propagation algorithms for arc consistency.

A difficulty in generalizing CSP algorithms to continuous domains is the formulation of the propagation rule, which involves many pathological cases. This paper presents a general and simple algorithm which subsumes specialized rules formulated for particular cases. The algorithm for binary constraints given in this paper is also sufficient to deal with constraints of more than two variables when all but two of them are restricted to precise values, as is the case in many practical problems. However, generalization to constraints involving more than two variables with interval labels requires explicit computation of the restricted feasible regions, as Lemma 1 does not hold for hypersurfaces in more than 2 dimensions.

Hyvönen [4] provides more powerful algorithms than Davis for the case of equality constraints. In particular, he addresses the numerical computations required in propagation and provides methods for achieving global consistency. However, I suspect that when generalized to inequalities and multiple constraints between pairs of variables, his algorithm would also require the formulation of total constraints as addressed in this paper to achieve local completeness.

I have further argued that the problem of looping is probably unavoidable considering the fact that the constraint satisfaction formalism can be used to formulate extremely difficult numerical analysis problems for which researchers have not found reliable algorithms. However, as I have shown, looping can only occur when the constraint network has cycles. An arc consistent labelling in the presence of cycles is not sufficient to guarantee a minimal network. Rather than trying to avoid infinite iterations, which is likely to be impossible, it seems more promising to compute labellings with higher degrees of consistency, such as path-consistency. So far, this possibility has been investigated for temporal constraint networks [2] and equality constraints [4].

Acknowledgements

I would like to thank an anomynous reviewer for suggesting several important improvements to the results of this paper. This work has been sponsored in part by the Swiss National Science Foundation, contract No. 20-32503.91.

References

[1] **E. Davis**: "Constraint Propagation with Interval Labels," *Artificial Intelligence* **32**, 1987

[2] **R. Dechter, I. Meiri, J. Pearl**: "Temporal Constraint Networks," *Artificial Intelligence* **49**, 1991

[3] **E. Freuder**: "A Sufficient Condition for Backtrack-free Search," *Journal of the ACM* **29**(1), 1982

[4] **E. Hyvönen**: "Constraint Reasoning based on Interval Arithmetic: the Tolerance Propagation Approach," *Artificial Intelligence* **58**, 1992

[5] **A. Mackworth**: "Consistency in Networks of Relations," *Artificial Intelligence* **8**, 1977

[6] **A. Mackworth, E. Freuder**: "The Complexity of some Polynomial Network Consistency Algorithms for Constraint Satisfaction,' *Artificial Intelligence* **25**, 1985

[7] **U. Montanari**: "Networks of Constraints: Fundamental Properties and Applications to Picture Processing," *Information Sciences* **7**, 1974

[8] **D. Waltz**: "Understanding Line Drawings of Scenes with Shadows," in P.H.Winston (ed.): *The Psychology of Computer Vision*, Mc-Graw-Hill, New York 1975