On the Complexity of Interval-Based Constraint Networks

Rony Shapiro[1], Yishai A. Feldman[2], and Rina Dechter[3]

September 19, 1998

Abstract

Acyclic constraint satisfaction problems with arithmetic constraints and domains consisting of sets of disjoint intervals have exponential complexity, since disjunctions of intervals may be introduced while propagating through the constraints. This has prompted many researchers to use approximations on the bounds of sets of intervals, resulting in sound, but incomplete, algorithms.

We delineate the complexity of propagation of sets of intervals through arithmetic constraints. For many types of constraint networks, our analysis shows linear, rather than exponential, complexity bounds. Furthermore, exponential complexity is a worst-case scenario that is surprisingly hard to achieve. In some cases, the number of disjoint intervals in the output of an acyclic constraint satisfaction problem is independent of the number of disjoint intervals in the input.

Some empirical results are presented, showing that the worst-case bound is not achieved for random intervals.

1 Introduction

Intuitively, when two or more sets of disjoint intervals are propagated through an aritmetic constraint, we expect the number of intervals in the result to be the product of the number of intervals in the inputs. Surprisingly, the number

[1]Department of Computer Science Tel-Aviv University (`rshapiro@idc.ac.il`).

[2]School of Computer and Media Sciences, The Interdisciplinary Center, Herzliya 46150, Israel (`yishai@idc.ac.il, http://www1.idc.ac.il/yishai`).

[3]Department of Information and Computer Science, University of Irvine, California (`dechter@ics.uci.edu`).

of *disjoint* intervals in the result is often much less. For example, given the intervals $A = [5, 25]$, $B = [28, 38]$, $C = [55, 60]$, and $D = [65, 80]$, the sum of the interval sets $\{A, B\} + \{C, D\}$ is the set of intervals

$$\{[60, 85], [70, 105], [83, 98], [93, 118]\}.$$

Note, however, that the intervals overlap, and are therefore equivalent to the *single* interval $[60, 118]$.

The rest of this paper elaborates on the above, investigating several constraints which can introduce disjunctions, and the behavior of simple combinations of such constraints. We also present some empirical results of arithmetic on sets of intervals, which are summarised in Table 1.

Constraints in Network	Worst-Case Complexity	Empirical Results
$+$	$O(\prod_{i=1}^{s} m_i)$	constant
$\times$	$O(\prod_{i=1}^{s} m_i)$	constant
$\|x\|$ and x^2	$O(2m)$	linear
Distance	$O(2m)$	linear
$+, \times$	$O(m\mathrm{Fib}(n+1))$	constant
$+$, x^2 or $\|x\|$	$O(m2^n)$	constant
$\times$, x^2, $\|x\|$	$O(2mn)$	constant
$+$, Distance	$O(\prod_{i=1}^{s} m_i)$	linear
$\times$, Distance	$O(m\mathrm{Fib}(n+1))$	linear

m_i = the number of intervals in the i^{th} input variable
s = the number of input variables
n = the number of constraints in the network

Table 1: Complexity of interval propagation through networks of constraints

2 Propagating Interval Sets

We consider the following arithmetic constraints: addition, multiplication, absolute value, square, and distance. For each constraint type, we analyze the propagation of intervals through a single constraint, and through networks consisting entirely of constraints of a single type. We then analyze the complexity of acyclic networks of combinations of these constraints. (An acyclic network is a network with no sequence of connected nodes in which the start and end nodes are the same.) Table 1 summarises our results.

2.1 Arithmetic Constraints

In this section, we will examine the propagation of intervals through simple arithmentic constraints, and homogeneous combinations of such constraints. We will use definitions of interval arithmetic similar to those defined by Moore [5].

2.1.1 Addition Constraints

Constraint Properties: Given an addition constraint $A + B = C$, values can propagate as follows: $C \leftarrow A + B$, $B \leftarrow C - A$, or $A \leftarrow C - B$.

The sum of two intervals, $x + y$, is $[\underline{x} + \underline{y}, \overline{x} + \overline{y}]$. (If x is an interval, we denote its endpoints by $\underline{x}$ and $\overline{x}$. Thus, $x = [\underline{x}, \overline{x}]$.) Likewise, the difference of two intervals, $x - y$, is $[\underline{x} - \overline{y}, \overline{x} - \underline{y}]$. The sum of two interval sets, $X + Y$, is the set of of disjoint intervals derived from the sum of each pair of intervals $\{x + y \,|\, x \in X, y \in Y\}$ by merging overlapping intervals. The difference is similarly defined. The generalization to n arguments is straightforward, and the following discussion applies to n-variable addition and subtraction constraints.

For both addition and subtraction, the number of intervals in the output variable is limited by the product of the number of intervals in each of the input variables. Specifically, if each variable is a single interval, the output variable will also be a single interval. If the number of intervals in variable X_i is m_i, then the upper bound on the number of intervals in the output variable X_j is $\prod_{i \neq j} m_i$. This is an upper bound because we merge overlapping intervals. It is not difficult to prove, by induction, the following theorem:

Theorem 1 *For every $n, m > 0$, it is possible to construct interval sets A and B of n and m disjoint intervals, respectively, such that the interval set $A + B$ has $n \times m$ disjoint intervals.*

From this theorem we may conclude that the combinatorial explosion of intervals with a single addition constraint is always *possible*. However, given *random* and uniform (non-intersecting) sets of intervals, such an explosion is *improbable*, since the number of constraints that must hold between all of the intervals in both sets in order to reach the upper bound increases in quadratic proportion to the product of the number of disjoint intervals in the sets. In Section 3, we show some empirical results to that effect.

Combining Addition Constraints: Consider a constraint network consisting of n addition constraints and m variables. If each variable contains a single interval, then propagation in any direction will not create a new interval ($1 \times 1 \times \ldots \times 1 = 1$). Similarly, if only a single variable has more than one interval, say $k > 1$, then after propagation, there will be at most k intervals in each variable. It is not hard to prove by induction (using Theorem 1) the following theorem:

Theorem 2 *In any acyclic constraint network consisting of addition constraints and variables $x_1, \ldots, x_m$, with the interval set corresponding to x_i consisting initially of s_i disjoint intervals, the upper bound on the number of intervals in any variable after all the intervals have been propagated is $\prod_{i=1}^{m} s_i$. Furthermore, this bound is reachable — it is possible to construct such a network.*

2.1.2 Multiplication Constraints

Constraint Properties: Given a multiplication constraint $A \times B = C$, intervals can propagate as follows.

If the interval sets A and B are given, C may be calculated by calculating the interval product of each pair of intervals from A and B, and deriving the disjoint interval set. The interval product of a pair of intervals is $a \times b = [\min(\underline{a}\underline{b}, \underline{a}\overline{b}, \overline{a}\underline{b}, \overline{a}\overline{b}), \max(\underline{a}\underline{b}, \underline{a}\overline{b}, \overline{a}\underline{b}, \overline{a}\overline{b})]$.

The number of intervals in the product C is bounded by the product of the number of intervals in the multiplicands. This bound is reachable — the proof (by construction) is similar to that of Theorem 1.

Note that the product of an interval that contains zero with another interval will always contain zero, hence all such interval products will overlap. This means that we may treat products interval sets that are wholly positive (or wholly negative) differently from products interval sets with intervals that may contain zeroes.

Theorem 3 *Given two interval sets, A and B, with n and m disjoint intervals, respectively, the upper bound on the number of intervals in their product is nm if none of the multiplicand intervals contain zero, $(n-1)m+1$ if an interval in A contains zero, $n(m-1)+1$ if an interval in B contains zero, or $(m-1)(n-1)+1$ if both do. In particular, if one multiplicand has any* *number of disjoint intervals, and the other has a single interval that contains zero, the product will be a* single *interval.*

The proof is by induction, analogous to that of Theorem 1, except that if an interval $a' \in A$ contains zero, then the product of any interval $b \in B$ with a' will also contain zero. Therefore, all of the products $a'b$ overlap, resulting in a single interval.

If the interval sets A and C are given, with n and m intervals, respectively, then for each pair of intervals $a \in A$, $c \in C$, we need to check what the possible values of b are (the roles of A and B are interchangeable in this discussion).

- If neither c nor a contain 0, then b is the quotient $c \div a = c \times [1/\overline{a}, 1/\underline{a}]$. The number of intervals in the quotient will be bounded by mn. It is easy to show that this bound is reachable by construction.
- If c contains zero, but a does not, then the above also applies. Note that the resulting b also contains zero. The bound on the number of intervals, according to Theorem 3, is then $(m-1)n+1$.
- If c does not contain zero, but a does, then we can exclude $\{0\}$ from a, since we know that the product of a and b is non-zero. This implies that b cannot contain zero as well. The result is two (open) intervals. Since any two quotients with a divisor containing zero result in two pairs of overlapping intervals, the upper bound on the number of intervals in the quotient in this case is $n(m-1)+2$.
- If both c and a contain zero, then we have no information on b to propagate.

Combining Multiplication Constraints: In a network consisting only of multiplication constraints, there may be a path through which a chain of divisions will occur while propagating intervals. Since each division by an interval containing zero will create two disjoint intervals, we would expect an exponential number of disjoint intervals as the output of the division chain. But, if we have a disjunction as a result of division, neither of the intervals in the disjunction will contain zero, and therefore, using them in a divisor will not result in two more disjunctions. If we multiply an open interval by an interval that contains zero, then we will get an interval that also contains zero, but, as we have seen in Theorem 3, both open intervals will be transformed into an interval that contains zero (this will be the open interval $(-\infty, \infty)$), so once again, the result is not exponential.

The complexity of propagating sets of intervals through a graph of multiplication constraints is the same as that of a graph of addition constraints, that is, $\prod_{i=1}^{m} s_i$, where m is the number of interval sets, and s_i is the number of intervals in the ith set. This is because a single multiplication constraint is similar to an addition constraint with regard to interval set propagation (by Theorems 1 and 3). Therefore, Theorem 2 holds for multiplication constraints as well as for addition.

2.1.3 Absolute Value and Square

Constraint Properties (Absolute Value): The absolute value of an interval is:

$$|x| = \begin{cases} [0, \max(|\underline{x}|, |\overline{x}|)] & \text{if } 0 \in x \\ x & \text{if } \underline{x} > 0 \\ [-\overline{x}, -\underline{x}] & \text{if } \overline{x} < 0 \end{cases}$$

Given an interval, we can uniquely find its absolute value. Given an absolute value, though, the information we can deduce about the interval is ambiguous, since there are potentially an infinite number of intervals whose absolute value is the given one. For example, if the absolute value of an interval is $[0, a]$, then the interval is either $[-a, b]$, where $0 \leq b \leq a$, or $[b, a]$, where $-a \leq b \leq 0$. The largest possible interval with an absolute value of $[0, a]$, however, is $[-a, a]$, so we must propagate this value. If A and B are sets of intervals, with m and n intervals, respectively, $A = |B|$, and B is given, with $\underline{b} > 0$ for all $b \in B$ then $m = 2n$, since the absolute value of each interval b is $[\underline{b}, \overline{b}] \cup [-\overline{b}, -\underline{b}]$, that is, there are two disjoint intervals in A for each interval in B. Thus, if the intervals in B are disjoint, then so are those in A.

Constraint Properties (Square): Unlike arithmetic over $\Re$, x^2 is not always equal to $x \cdot x$. For example, $[-5, 3] \times [-5, 3] = [-15, 25]$. But, $[-5, 3]^2 = \{x^2 | x \in [-5, 3]\} = [0, 25]$. This is a good reason for treating the square constraint separately.

In general, given an interval $x = [\underline{x}, \overline{x}]$:

$$x^2 = \begin{cases} [0, \max(|\underline{x}|, |\overline{x}|)^2] & \text{if } 0 \in x \\ [\min(|\underline{x}|, |\overline{x}|)^2, \max(|\underline{x}|, |\overline{x}|)^2] & \text{otherwise} \end{cases}$$

As with the absolute value constraint, this can potentially reduce the number of disjoint intervals passing through it in this direction. For example, $[-5, -3]$ and $[3, 5]$ are disjoint, but their squares coincide.

Given $y = [\underline{y}, \overline{y}]$, with the constraint that $y \geq 0$, x could be a union:

$$x = \begin{cases} [-\sqrt{\overline{y}}, -\sqrt{\underline{y}}] \vee [\sqrt{\underline{y}}, \sqrt{\overline{y}}] & \text{if } \underline{y} > 0 \\ [-\sqrt{\overline{y}}, \sqrt{\overline{y}}] & \text{if } \underline{y} = 0 \end{cases}$$

This means that for $A = B^2$ with B given, there are (at most) two disjoint intervals in A for each interval in B.

Combining Absolute Value or Square Constraints: Since these constraints are binary, the only way to combine them is by chaining. The number of intervals in the output of such a chain cannot be more than twice the number of intervals in the input, since a negative interval that may have been created by one absolute value or square constraint will not be propagated through the next, by the constraint's definition.

2.1.4 Distance

Constraint Properties: The distance constraint is of the form

$$d = \|(p_x, p_y), (q_x, q_y)\|$$

This constraint defines the Euclidean distance between two points on a plane; d is always non-negative. The other variables represent the coordinates of two points on the plain, and are unrestricted. This constraint may be represented as a network of the addition and square constraints discussed above: $p_x + w = q_x; p_y + h = q_y; W = (w)^2; H = (h)^2; W + H = D; d^2 = D$. This network represents the equation $d = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$. Such a network, of addition and square constraints, has exponential complexity (see Section 2.2.2). If we treat the distance contraint as a primitve constraint, however, the resulting complexity is linear in the number of intervals.

As noted in Section 2.1.3, a union in the square root of an interval exists iff the lower bound of the interval is greater than zero. In the context of the distance constraint, this means that if we set a non-zero minimum on the distance ($d > 0$), then at least one of addends W or H must have a lower bound greater than zero (because they cannot be negative), therefore one or both of the square roots (w, h) must be a union of intervals. In general, n disjoint intervals constraints on d will propagate to $2n$ intervals in a coordinate variable if none of the intervals contain zero, and $2n - 1$ disjoint intervals if one of d's intervals contain zero.

If we consider the problem geometrically, as a circle centered around (p_x, p_y) with radius d, then it is obvious that the effect of restricting the values of of a single coordinate variable, say q_x, to an interval [a,b] is to define a vertical "strip" of the circle, or $\emptyset$ if $a > p_x + d$ or $b < p_x - d$.

Combining Distance Constraints: Constraining the distance variable to n disjoint constraints results in at most $2n$ disjoint interval constraints on the other variables. If there are m distance constraints so connected, there will be $2nm$ intervals propagated. If a coordinate variable is shared, no disjunctions are introduced, and interval constraints on the shared variable propagate without change to the other variables. A tree of distance constraints may be created, with each non-leaf coordinate variable being the distance variable of another distance constraint. The complexity in this case is as that of an addition tree, that is, the product of the number of interval sets in the coordinate variables.

2.2 Mixing Constraints

In the following sections, we discuss the effects of mixing constraints of different types on the complexity of the propagated interval sets.

For each pair of the constraints discussed above, we will see under which circumstances we may get an exponential explosion of interval sets, given, initially, single intervals at all the variables. We want to find both the largest problem that is tractable, and the smallest that is not.

2.2.1 Addition and Multiplication

As we have seen in Sections 2.1.1 and 2.1.2, acyclic networks consisting only of addition constraints or only of multiplication constraints are tractable when the variables are all initially single intervals (addition cannot cause intervals to split, whereas the splits possibly introduced by division are self-bounding).

When addition and multiplication constraints are combined, the number of intervals may increase exponentially. Let $(-\infty, -q], [q, \infty)$ be the quotient of two intervals propagated by a multiplication constraint. We will constrain the two intervals by a single interval, $[-b, b]$, with $b > q$, such that we now have two disjoint, but closed, intervals, one wholly positive, the other wholly negative: $[-b, -q], [q, b]$. By using an addition constraint to add a constant value to intervals, we can "shift" them, causing one of them to straddle zero. For example, we could add the interval $[(b + q)/2, (b + q)/2]$. The sum with the negative interval would be $[(q-b)/2, (b-q)/2]$, which contains zero. The sum with the positive interval would remain wholly positive. If we propagate this pair through another multiplication constraint as a quotient, using each interval in turn as the divisor of a third interval, we will have another split, that is, three intervals in the new quotient.

For example, the quotient $[5,5] \div [-1,1]$ is $(-\infty,-5], [5,\infty)$. Intersecting these intervals with $[-10,10]$ results in $[-10,-5], [5,10]$. Adding the interval $[7,7]$ to this pair gives $[-3,2], [12,17]$. Note that one interval contains zero. We now use this pair of intervals as the divisor of a quotient, say $[1,1] \div \{[-3,2],[12,17]\}$. The result is three disjoint intervals: $(-\infty,-1/3], [1/2,\infty)$, and $[1/17,1/12]$. The process can be repeated.

Thus, by chaining n pairs of addition, intersection and multiplication constraints, one could get exactly $\mathrm{Fib}(n+1)$ intervals, as follows. After the first multiplication, there may be two intervals, one wholly negative, the other wholly positive. We will mark this as 1/1. If we add an appropriate constant, one of the intervals will now straddle zero, so multiplying the two segments by a third will now result in one segment above zero, and two below (the new one and the one from the previous multiplication), i.e. 2/1. Now, by adding another constant, we can get one of the lower pair to straddle zero, so that multiplication will split two intervals, resulting in five intervals: 2/3. At each step, one side has the number of intervals on the other side added to it, with the total number of intervals being the sum of intervals on both sides of zero. This is, for n stages, $\mathrm{Fib}(n+1)$, which is exponential.

2.2.2 Addition and Absolute Value or Square

A network of alternating addition and absolute value constraints that results in 2^n intervals after n pairs of constraints, may be created as follows:

The absolute value constraints will propagate intervals in the "backward" direction, that is, given an absolute value, it returns the intervals which produced it. As we have seen in Section 2.1.3, this results in a union of intervals, one wholly positive, the other wholly negative. Using an addition constraint, we may add an interval large enough so that the sum union of intervals is wholly positive (and still disjoint): If the interval pair is $\{[-b,-a],[a,b]\}$, we can add the interval $[c,c]$, where $c > b$. This would result in $\{[(c-b),(c-a)],[(c+a),(c+b)]\}$, which is wholly positive. Feeding this pair into an absolute value constraint would result in four disjoint intervals, and we can repeat the process.

With addition and square constraints, one could create a chain similar to that described in Section 2.2.1, alternating addition and square roots. The addition constraints could shift the intervals $\{[-\sqrt{\overline{Y}}, -\sqrt{\underline{Y}}], [\sqrt{\underline{Y}}, \sqrt{\overline{Y}}]\}$ so that both will be fully positive (by adding $k > |\sqrt{\overline{Y}}|$), so that the resultant intervals would both be wholly positive input for another square root constraint, resulting in four intervals. This could be continued indefinitely (assuming infinite precision arithmetic), giving, after n stages, 2^n intervals.

2.2.3 Multiplication and Absolute Value or Square

In Section 2.2.2, we used the addition constraint to "shift" a wholly positive and wholly negative pair of interval sets to be wholly positive, thus enabling a doubling of the intervals in the next absolute value constraint. This method is not possible with multiplication constraints, since multiplication (and division) is symmetrical regarding sign. Therefore, multiplication and absolute value constraints together can result in no more than twice as many disjoint intervals as multiplication alone, that is, $2\prod_{i=1}^{m} s_i$, where m is the number of interval sets, and s_i is the number of intervals in the ith set, as shown in Section 2.1.2.

The square root of an interval is a union of two intervals, one wholly positive and the other wholly negative, of the form $\{[-b, -a], [a, b]\}$, where a and b are the respective square roots of the interval $[a^2, b^2]$. Due to the identity of the form of the union of intervals with that of the form of union of intervals of the absolute value constraint, the arguments and conclusions describing multiplication and absolute value apply to the combination of multiplication and square.

2.2.4 Distance and Other Constraints

As discussed in Section 2.1.4, the propagation of intervals through a distance constraint may be compared to the propagation through an addition constraint, unless the distance variable is the input, and the interval sets propagate to the coordinate variables. In this case, the number of intervals may be twice the number of interval constraints on the distance variable. Therefore, a constraint network consisting of distance constraints and addition constraints will have the same properties as a network consisting wholly of addition constraints. Similarly, a constraint network comprised of multiplication and distance constraints will have the same properties as a network of multiplication and addition constraints.

3 Empirical Results

In this section, we will present some results of propagating sets of intervals through interval extensions of arithmetic operations. All the results presented here were generated by a C++ program using double precision arithmetic and the standard C library random number generator. Each data point is the average of ten trials.

3.1 Sum and Product of Two Interval Sets

In Figure 1 we see the number of disjoint intervals in the sum and product of two disjoint interval sets, where the average width of the intervals in the input is $1,000$, and the range of endpoint values is restricted to $\pm 50,000$. The number of intervals in both input sets is equal, and is varied from 1 to 60. When there are few intervals in the input, the number of disjoint intervals in the output increases as expected, that is, as the product of the number of intervals in the input. When the number of intervals in the input increases, however, the number of disjoint intervals in the result decreases. Another

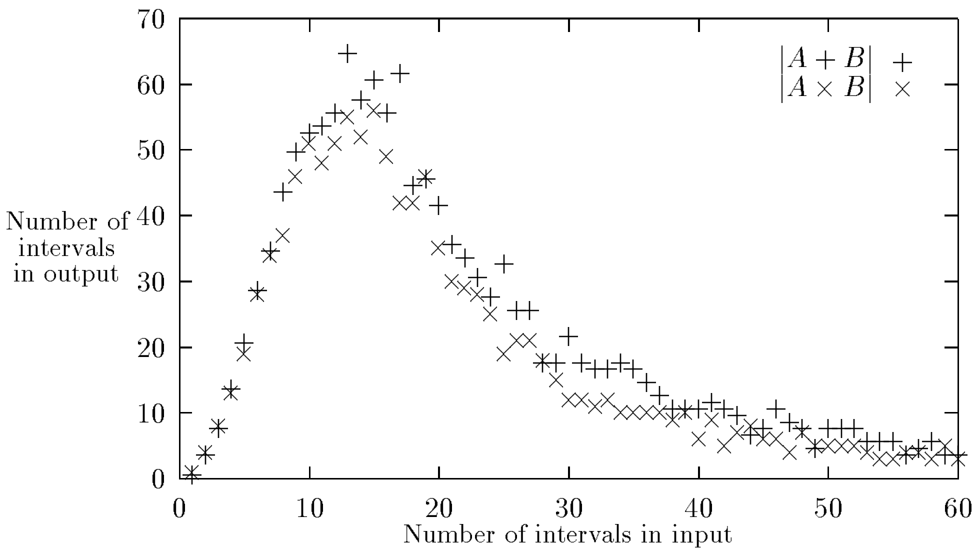

Figure 1: Number of intervals in sum and product of two interval sets

interesting point about these results is that the number of intervals in the product tends to be less than the number of intervals in the sum, for the same input. This is because the number of disjoint intervals in a product where a multiplicand interval contains zero will be less than the corresponding sum, as explained in Section 2.1.2.

3.2 Sum and Product of Four Interval Sets

We show the result of sums and product of four input interval sets. The conditions are the same as described in the previous section, i.e., the average width of the intervals in the input is $1,000$, and the range of endpoint values is restricted to $\pm 50,000$. The number of intervals in both input sets is equal, and is varied from 1 to 30. The result is shown in Figure 2. As may be expected, the number of disjoint intervals in the result decreases much more sharply than when the input consists of only two interval sets.

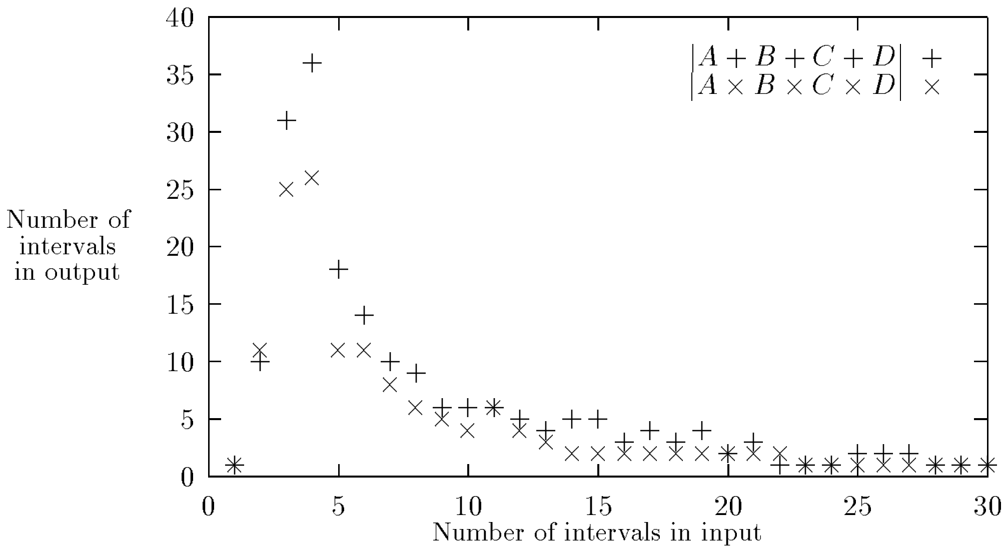

Figure 2: Number of intervals in sum and product of four interval sets

4 Related Work

Sam-Haroud and Faltings [6] developed algorithms for solving constraint systems with continuous domains and relations defined by sets of algebraic or transcendental constraints that are polynomial-time for a large set of problems.

Temporal constraint networks extend the network-based methods of constraint satisfaction to continuous variables. Dechter, Meiri, and Pearl [2] define *temporal constraint satisfaction problems* (TCSP) as follows: Variables $V_1, \ldots, V_n$ have continuous domains. The constraints are either *unary,* restricting the domain of a variable to a given set of (disjoint) intervals, or *binary*, restricting the distance $X_i - X_j$ between two variables to a given set of (disjoint) intervals. They show that the problem is easy when the set of intervals is restricted to a single interval.

Schwalb and Dechter [9] show that the complexity of enforcing path consistency on TCSPs is exponential, due to the fragmentation of intervals into subintervals. They present algorithms that bound the fragmentation by computing looser constraints that subsume the subintervals (at the cost of losing precision).

Koubarakis [3] discusses a tractable special case of the problem. He shows that strong $(2V + 1)$-consistency is necessary and sufficient for global consistency, where V is the maximum number of variables in any disjunction of inequations. In a later work [4], the class of constraints is extended to include disjunctions of inequations with *at most one* inequality per disjunction.

Graphic editors are applications that can incorporate constraint networks in a useful manner. In such applications, there is usually a constrint network with a given solution, which is *perturbed,* either by adding a constraint, or by changing the value of one or more of the variables. The goal is to find a new solution for the perturbed network.

DeltaBlue [8] and SkyBlue [7] are constraint engines for graphic editors that support linear equalities in an efficient manner, as long as the constraint graph is acyclic. SkyBlue supports cycles in constraint graphs indirectly, by detecting them and calling an external mechanism, such as a simultaneous linear equation solver. Recently, Indigo [1] has extended the class of constraints to include linear inequalities.

5 Conclusions and Future Work

In this paper, we have analyzed the complexity of propagating sets of intervals through algebraic constraints. The results of the analysis are applicable to the analysis of the trade-offs between the expressive power of a constraint engine and the complexity involved.

From our analysis of arithmetic constraints, we see that if no interval sets are introduced in the constraint network, then none will be generated during propagation in the following networks:

- Only addition and unary (domain) constraints.
- Only multiplication and unary constraints, if the domain is restricted to wholly positive numbers (without zero).
- Algebraic constraint networks in which the intervals will not contain zero in the course of propagation. Zeroes may be introduced, for example, by the difference between two wholly positive intervals.

Throughout this paper, we have considered only acyclic constraint networks. There are many practical problems, however, which map to constraint networks with cycles. Investigating the actual limits of interval complexity in such networks might reveal some surprises.

References

[1] A. Borning, R. Anderson, and B. Freeman-Benson. The Indigo algorithm. Technical Report 96-05-01, Department of Computer Science and Engineering, University of Washington, July 1996.

[2] R. Decther, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.

[3] M. Koubarakis. From local to global consistency in temporal constraint networks. In *Proc. 1st Int. Conf. Principles and Practice of Constraint Programming, LNCS 976*, pages 53–69, Sept. 1995.

[4] M. Koubarakis. Tractable disjunctions of linear constraints: Basic results and applications to temporal reasoning. In *Proc. 2nd Int. Conf. Principles and Practice of Constraint Programming, LNCS 1118*, pages 297–307, Aug. 1996.

[5] R. E. Moore. *Methods and Applications of Interval Analysis.* SIAM, 1979.

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

[7] M. Sannella. SkyBlue: A multi-way local propagation constraint solver for user interface construction. In *Proc. 1994 ACM Symp. User Interface Software and Technology*, pages 137–146, 1994.

[8] M. Sannella, J. Maloney, B. Freeman-Benson, and A. Borning. Multi-way versus one-way constraints in user interfaces: Experience with the DeltaBlue algorithm. *Software Practice and Experience*, 23(5):529–566, May 1993.

[9] E. Schwalb and R. Dechter. Processing disjunctions in temporal constraint networks. *Artificial Intelligence*, 93:29–61, 1997.