Exponential Propagation for Set Variables

Justin Yip and Pascal Van Hentenryck

Brown University, Box 1910, Providence, RI 02912, USA

Abstract. Research on constraint propagation has primarily focused on designing polynomial-time propagators sometimes at the cost of a weaker filtering. Interestingly, the evolution of constraint programming over sets have been diametrically different. The domain representations are becoming increasingly expensive computationally and theoretical results appear to question the wisdom of these research directions. This paper explores this apparent contradiction by pursuing even more complexity in the domain representation and the filtering algorithms. It shows that the product of the length-lex and subset-bound domains improves filtering and produces orders of magnitude improvements over existing approaches on standard benchmarks. Moreover, the paper proposes exponential-time algorithms for NP-hard intersection constraints and demonstrates that they bring significant performance improvements and speeds up constraint propagation considerably.

1 Introduction

Constraint programming (CP) is often seen as a computational methodology for tackling constraint satisfaction problems (CSPs) characterized by the slogan

$$\text{Constraint Programming} = \text{Filtering} + \text{Search}.$$

Since most CSPs are NP-complete, CP uses filtering algorithms and constraint propagation to reduce the variable domains and hence the search tree to explore. The hope is that the reduction in the search space is sufficient to solve problems of interest in reasonable time. In general, researchers have focused on designing polynomial-time algorithms for filtering, leaving the potentially exponential behavior in the search component. There are exceptions of course, and we will review some of them later, but researchers overwhelmingly focus on polynomial-time filtering algorithms, sometimes at the expenses of enforcing arc or bound consistency. This paper takes the other road and argues that exponential filtering algorithms and constraint propagation may be highly beneficial in practice. It is motivated by the fact that reasonable exponential behavior in the filtering algorithm may produce significant reduction of search space and can therefore be cost-effective. Moreover, such a reasonable exponential behavior has beneficial effects on constraint propagation allowing further reduction of the search space, an observation made by Bessiere and Régin in [1] where they solve CSPs on the fly to achieve arc consistency on a global constraint. Finally, since the overall approach is exponential in the worst case, it may be preferable to shift

some of the exponential behavior from the largely agnostic search to the filtering component where we can exploit the semantics of the constraints and locality.

This paper evaluates this idea in the context of CSPs over set variables. Set variables are natural objects for modeling classes of combinatorial problems but they raise fundamental computational issues. Since their domains often represent an exponential number of sets, maintaining an explicit enumeration of the domain values is computationally expensive and hence much effort has been devoted to approximating a set domain or using a compact and precise representation. The subset-bound domain was introduced in the early 90s and is now available in many CP-solvers (e.g., [2, 3]). It maintains two sets r, p and defines its domain as $\{s \mid r \subseteq s \subseteq p\}$. The subset-bound domain supports polynomial-time filtering algorithms for a variety of constraints but the pruning it offers is generally weak.[1] As a result, much research has been devoted to design richer representations of set domains. Sometimes the subset-bound domain is enhanced with a cardinality component to restrict the cardinality of the set variable [4, 5], which often results in much stronger propagation. However, Bessiere and al. in [6] have shown that some natural global constraints become intractable when moving to this domain. Some exact representations of sets use ordered binary decision diagrams and their propagation algorithms can obviously take exponential time[7]. The length-lex representation takes a dual perspective by primarily capturing the cardinality and lexicographical information [8]. A length-lex domain is defined by two bounds l, u as $\{s \mid l \preceq s \preceq u\}$ where $\preceq$ is the (total) length-lex ordering. Here the filtering algorithms for many elementary constraints are typically polynomial-time but the constraint-propagation algorithm may take exponential time to converge to a fixpoint [9].[2] There is thus an abundance of negative theoretical results on richer set representations. Yet, on a wide variety of standard benchmarks, the richer representations bring orders of magnitude improvements compared to the subset-bound domain or traditional encodings in terms of finite-domain variables[7, 14].

The goal of this paper is thus to explore whether it is beneficial to boost constraint propagation over set variables even further. It explores two ideas: the product of the length-lex and subset-bound domains which has never been evaluated before and exponential propagators for intersection constraints. Its main contributions are as follows:

1. It shows that the propagation of simple unary constraints in the length-lex domain may take exponential time to converge, yet the length-lex domain brings orders of magnitude improvements over other representations.
2. It demonstrates that the product of the subset-bound and length-lex domains brings additional pruning on intersection constraints and significant improvements in efficiency.

[1] Note that, even with this compact representation, many intractable problems can be encoded by a unary constraint over a single set variable.

[2] The fact that constraint-propagation algorithms may take exponential time to converge is not specific to set variables. It appears for instance in numerical continuous CSPs (e.g., [12]) and the propagation of cumulative constraints [13].

3. It proves the $W[1]$-Hardness and NP-completeness of unary intersection constraint for the subset-bound with cardinality and length-lex domains.
4. It proposes exponential but complete filterings for these intersection constraints and shows that they bring again an order of magnitude improvement in efficiency compared to existing approaches.
5. It shows that these exponential filtering algorithms speeds up the convergence of the constraint propagation algorithm considerably.

The rest of the paper is organized as follows. Section 2 reviews common set representations and formally specifies the consistency notion to establish the theoretical results of the paper. Section 3 discusses intractability issues for set variables. Section 4 contrasts the theoretical results with an experimental evaluation of the various domains. Section 5 proposes an exponential-time propagator for a $W[1]$-hard unary intersection and Section 6 evaluates the effectiveness and efficiency of the proposed constraint. Section 7 discusses some of the related work. Section 8 concludes the paper.

2 Domain Representations and Consistency Notions

This section reviews the domain representation for set variables and the consistency notions for set constraints.

Notations For simplicity, we assume that sets take their values in a universe $U(n)$ of integers $\{1, \ldots, n\}$ equipped with traditional set operations. n, m are integers denoting the size of a universe, the number of variables, or both. X, possibly subscripted, is a set variable. Elements of $U(n)$ are denoted by the letter e. Sets are denoted by l, u, s, r, p. A subset s of $U(n)$ of cardinality c is called c-set and is denoted as $\{s_1, s_2, ..., s_c\}$ where $s_1 < s_2 < ... < s_c$. The length-lex ordering $\preceq$, proposed in [8], totally orders sets first by cardinality and then lexicographically.

Definition 1 (Length-Lex Ordering). *The length-lex ordering is defined by*

$$s \preceq t \textit{ iff } s = \emptyset \vee |s| < |t| \ \vee |s| = |t| \ \wedge (s_1 < t_1 \vee s_1 = t_1 \wedge s \setminus \{s_1\} \preceq t \setminus \{t_1\})$$

Its strict version is defined by $s \prec t$ *iff* $s \preceq t \wedge s \neq t$.

Example 1. Given $U(4) = \{1, \ldots, 4\}$, we have $\emptyset \prec \{1\} \prec \{2\} \prec \{3\} \prec \{4\} \prec \{1,2\} \prec \{1,3\} \prec \{1,4\} \prec \{2,3\} \prec \{2,4\} \prec \{3,4\} \prec \{1,2,3\} \prec \{1,2,4\} \prec \{1,3,4\} \prec \{2,3,4\} \prec \{1,2,3,4\}$.

Definition 2 (ll-domain). *A length-lex domain (ll-domain) is a pair of sets* $ll\langle l, u, n\rangle$. *It contains all sets (inclusively) in the universe* $U(n)$ *between* l *and* u *in the length-lex ordering:*

$$ll\langle l, u, n\rangle \equiv \{s \subseteq U(n) | l \preceq s \preceq u\}.$$

Example 2. The length-lex domain $ll\langle\{1,3,8\}, \{1,5,8\}, 8\rangle$ denotes the set $\{\{1,3,8\}, \{1,4,5\}, \{1,4,6\}, \{1,4,7\}, \{1,4,8\}, \{1,5,6\}, \{1,5,7\}, \{1,5,8\}\}$.

Definition 3 (ll-bound consistency). *A set constraint $\mathcal{C}(X_1, ..., X_m)$ (X_i are set variables using the ll-domain) is said to be ll-bound consistent if and only if $\forall 1 \leq i \leq m$,*

$$\begin{aligned} & \exists x_1 \in d(X_1), ..., x_m \in d(X_m) \text{ s.t. } C(x_1, .., x_{i-1}, l_{X_i}, x_{i+1}, .., x_m) \\ \wedge\ & \exists x_1 \in d(X_1), ..., x_m \in d(X_m) \text{ s.t. } C(x_1, .., x_{i-1}, u_{X_i}, x_{i+1}, .., x_m) \end{aligned}$$

where $d(X_i) = ll\langle l_{X_i}, u_{X_i}, n_{X_i}\rangle$ denotes the domain of X_i.

Enforcing ll-bound consistency assures that the bounds of each variable appears in a solution to the constraint, which corresponds to the traditional notion of bound consistency on finite-domain variables. It is a rather strong property for set domains and is well-defined because the length-lex ordering is total.

Definition 4 (sbc-domain). *A subset-bound + cardinality domain (sbc-domain) $sbc\langle r, p, \check{c}, \hat{c}\rangle$ consists of a required set r and a possible set p, a minimum and maximum cardinalities $\check{c}$ and $\hat{c}$, and represents the set of sets*

$$sbc\langle r, p, \check{c}, \hat{c}\rangle \equiv \left\{ s \mid r \subseteq s \subseteq p \wedge \check{c} \leq |s| \leq \hat{c} \right\}$$

Example 3. The sbc-domain $sbc\langle\{1\}, \{1, 3, 4, 5, 7, 8\}, 3, 3\rangle$ denotes the set $\{\{1, 3, 4\}, \{1, 3, 5\}, \{1, 3, 7\}, \{1, 3, 8\}, \{1, 4, 5\}, \{1, 4, 7\}, \{1, 4, 8\}, \{1, 5, 7\}, \{1, 5, 8\}, \{1, 7, 8\}\}$.

Definition 5 (sbc-bound consistency). *A set constraint $\mathcal{C}(X_1, ..., X_m)$ (X_i are set variables using the sbc-domain) is said to be sbc-bound consistent if and only if $\forall 1 \leq i \leq m$,*

$$\begin{aligned} & \exists x_1 \in d(X_1), ..., x_m \in d(X_m) \text{ s.t. } C(x_1, ..., x_m) \\ \wedge\ & r_{X_i} = \bigcap_{\forall 1 \leq j \leq m, x_j \in d(X_j) : C(x_1, ..., x_m)} x_i \wedge p_{X_i} = \bigcup_{\forall 1 \leq j \leq m, x_j \in d(X_j) : C(x_1, ..., x_m)} x_i \\ \wedge\ & \exists x_1 \in d(X_1), ..., x_m \in d(X_m) \text{ s.t. } |x_i| = c_{\check{X}_i} \wedge C(x_1, ..., x_m) \\ \wedge\ & \exists x_1 \in d(X_1), ..., x_m \in d(X_m) \text{ s.t. } |x_i| = c_{\hat{X}_i} \wedge C(x_1, ..., x_m) \end{aligned}$$

where $d(X_i) = sbc\langle r_{X_i}, p_{X_i}, c_{\check{X}_i}, c_{\hat{X}_i}\rangle$ denotes the domain of X_i.

Note that this definition requires the constraint to have a solution consistent with the domains of the variables (first condition) and to have solutions consistent with each of the cardinality bounds of the variables (fourth and fifth conditions). We now define the product of the length-lex and subset-bound domain. (Similar hybrid domain representations were proposed in [10, 11].)

Definition 6 (ls-domain). *A length-lex + subset-bound domain (ls-domain) is the intersection of the two aforementioned domains. A ls-domain $ls\langle l, u, n, r, p\rangle$ consists of two bounds l, u for the length-lex ordering, a universe size n, a required set r, and a possible set p. It represents the set of sets*

$$ls\langle l, u, n, r, p\rangle \equiv ll\langle l, u, n\rangle \cap sbc\langle r, p, |l|, |u|\rangle$$

Example 4. The ls-domain $ls\langle\{1,3,8\},\{1,5,8\},8,\{1\},\{1,3,4,5,7,8\}\rangle$ (which is the intersection of domains in Example 2 and 3) denotes the set $\{\{1,3,8\},\{1,4,5\},\{1,4,7\},\{1,4,8\},\{1,5,7\},\{1,5,8\}\}$.

Definition 7 (ls-bound consistency). *A set constraint $\mathcal{C}(X_1,...,X_m)$ (X_i are set variables using the ls-domain) is said to be ls-bound consistent if and only if $\forall 1 \leq i \leq m$,*

$$\begin{aligned} & l_{X_i} \in d(X_i) \wedge \exists x_1 \in d(X_1),...,x_m \in d(X_m) : C(x_1,..,x_{i-1},l_{X_i},x_{i+1},..,x_m) \\ \wedge\ & u_{X_i} \in d(X_i) \wedge \exists x_1 \in d(X_1),...,x_m \in d(X_m) : C(x_1,..,x_{i-1},u_{X_i},x_{i+1},..,x_m) \\ \wedge\ & r_{X_i} = \bigcap_{\forall 1\leq j\leq m, x_j\in d(X_j):C(x_1,...,x_m)} x_i \wedge p_{X_i} = \bigcup_{\forall 1\leq j\leq m, x_j\in d(X_j):C(x_1,...,x_m)} x_i \end{aligned}$$

where $d(X_i) = ls\langle l_{X_i}, u_{X_i}, n_{X_i}, r_{X_i}, p_{X_i}\rangle$

We now show that the ls-domain is strictly stronger than the conjunction of the ll-domain and sbc-domain.

Lemma 1. *Enforcing bound consistency on a ls-domain is strictly stronger than enforcing bound consistency separately on the decomposition of the ll-domain and sbc-domain.*

Proof. Clearly enforcing bound consistency on the ls-domain is at least as strong. Consider a unary constraint $|X \cap \{4,5,7\}| \leq 1$ and the ls-domain in Example 4. It is bound-consistent for the decomposition, since the lower and upper bounds satisfy the constraint and the required and possible sets are bound-consistent. However, for the ls-domain, only three domain values, i.e., $\{1,3,8\},\{1,4,8\},\{1,5,8\}$, satisfy the constraint. Element 8 belongs to all solutions and thus to the required set. Enforcing bound consistency for the ls-domain yields $ls\langle\{1,3,8\},\{1,5,8\},8,\{1,8\},\{1,3,4,5,7,8\}\rangle$. □

3 Theoretical Results on Intersection Constraints

We now present a number of theoretical results, which shed light on the behavior and complexity of filtering algorithms and constraint propagation on set domains. In the following, we use $bc_\theta\langle\mathcal{C}\rangle$ to denote a bound-consistency propagator and $hs_\theta\langle\mathcal{C}\rangle$ to denote a feasibility routine for constraint $\mathcal{C}$ on a θ-domain.

The first result we mentioned is well-known but quite interesting and concerns the sbc-domain.

Theorem 1. $hs_{sbc}\langle|X_i \cap X_j| \leq 1, \forall i < j\rangle$ *is NP-hard. [6]*

We consider a special case of this constraint in which all but one variables are bounded. We show that, even in this simple unary case, enforcing bound consistency on both the sbc-domain and the ll-domain is fixed-parameter intractable.

Definition 8 (atmost1). $atmost1(\{s_1,..,s_m\},X) \equiv |X \cap s_i| \leq 1,\ \forall 1 \leq i \leq m$

Theorem 2. *$hs_{sbc}\langle atmost1(\{s_1,..,s_m\},X)\rangle$ is NP-hard.*

Proof. Reduction from k-Independent Set. Instance: Graph $G=(V,E)$ and a positive integer $k \leq |V|$. Question: Does G contains an independent set of size k, i.e. a k-subset V' of V such that no two vertices in V' join by an edge in E.

We construct an instance of CSP with one sbc-variable X and one constraint $atmost1(\{s_1,..,s_m\},X)$. Intuitively, X corresponds to a independent set and each set s_i corresponds to the neighborhood of vertex i and itself. Hence, X can take at most 1 element from each set corresponds to the restriction that no two vertices in the independent set join by an edge.

Formally, for every $i \in V$, $s_i = \{i\} \cup adj(i)$ (where $adj(i)$ denotes the neighborhood of vertex i), and $X \in sbc\langle\emptyset,V,k,k\rangle$. The CSP has a solution if and only if G has a independent set of size k. $\Rightarrow$ Given a k independent set V', we can construct a solution by setting $X=V'$ since every element in X actually corresponds to a vertex. When X takes an element i, since the size of intersection is at most 1, it cannot take any other element from set s_i (i.e. $adj(i)$), the definition of independent set guarantees this. $\Leftarrow$ Given a consistent assignment of X, it is a independent set since any edge corresponds to taking two element from the same set which violates the $atmost1$ constraint. □

Corollary 1. *$hs_{sbc}\langle atmost1(\{s_1,..,s_m\},X)\rangle$ is $W[1]$-hard.*

Proof. k-Independent Set is a $W[1]$-Complete problem.[15] □

Corollary 2. *$hs_{ll}\langle atmost1(\{s_1,..,s_m\},X)\rangle$ is $W[1]$-hard.*

Proof. For any sbc-domain that contains only all k-sets of some universe, there exists an equivalent ll-domain $sbc\langle\emptyset,V,k,k\rangle \equiv ll\langle\triangle_k,\triangledown_k,|V|\rangle$ with $\triangle_k = \min_{\preceq}\{s \mid s \subseteq V \wedge |s|=k\}$ and $\triangledown_k = \min_{\preceq}\{s \mid s \subseteq V \wedge |s|=k\}$. □

This result has an interesting corollary. Consider the propagation of a set of unary constraints of the form $|X \cap s_i| \leq 1$ $(1 \leq i \leq n)$. These constraints enjoy a polynomial-time bound-consistency algorithm in the length-lex domain. By definition of bound consistency, constraint propagation terminates in a failure or in a state where the bounds of the variable are solutions. Hence, by Corollary 2, constraint propagation cannot run in time $O(f(k)n^{O(1)})$ in the worst case.

Corollary 3. *The propagation algorithm for a collection of $bc_{ll}\langle|X \cap s_i| \leq 1\rangle$ over X cannot run in time $O(f(k)n^{O(1)})$ in the worst case unless $FPT = W[1]$.*

Similar results hold for other intersection constraints.

Definition 9 (exact1). $exact1(\{s_1,..,s_m\},X) \equiv |X \cap s_i| = 1,\ \forall 1 \leq i \leq m$

Theorem 3. *$hs_{sbc}\langle exact1(\{s_1,..,s_m\},X)\rangle$ is NP-hard.*

Proof. Reduction from 1-in-3 SAT. Instance: Set of n variables and m clauses, where each clauses consists of exactly three literals and each literal is either

a variable or its negation. Question: Does there exist a truth assignment to variables such that each clause has exactly one true literal?

Given a instance of 1-in-3 SAT, we construct a CSP with a $exact1$ constraint. A set variable X associated with a sbc-domain $sbc\langle\emptyset, \{1, -1, .., n, -n\}, n, n\rangle$ corresponds to a truth assignment. $i \in X$ means variable i is true and vice versa. There are two types of sets. Set $s_i = \{i, -i\}$ $(1 \leq i \leq n)$ ensures a variable can either be true or false. Set $t_j = \{p, -q, r\}$ corresponds to a clause $(x_p \vee \neg x_q \vee x_r)$ guarantees that exactly one of its literal is true. Hence, we post the constraint $exact1\langle\{s_1, .., s_n, t_1, .., t_m\}, X\rangle$. Clearly, the input instance has feasible assignment if and only if the CSP has a solution. □

Similarly, a NP-hardness proof for the feasibility routine of $atleast1$ is obtained from Theorem 3 by changing the input instance to 3SAT.

As mentioned earlier, the potentially exponential behavior of constraint propagation was pointed out in [9] for knapsack constraints and similar results exist for continuous constraints and edge-finding algorithms for cumulative constraints. What is somewhat surprising here is the simplicity of the constraint involved, which are simple unary intersection constraints. *This abundance of negative theoretical results may lead researchers to conclude that the sbc-domain and, even more so, the ll-domain are unworthy of any consideration. Experimental results however clearly indicate otherwise as we now discuss.*

4 Experimental Behavior of Domain Representations

The experimental results in this section compare the ll-domain and ls-domain, as well as the hybrid BDD-SAT recently proposed in [16] which provides state-of-the-art results in this area. The models use efficient complete filtering algorithms for binary constraints on these different domains [16–18, 14, 19]. We evaluate them on two standard CSP benchmarks: the social golfer problem and the steiner triple system. These models run on a 2.4GHz Core 2 Duo laptop with 4GB of memory. The time limit is 1800 seconds and × indicates a timeout. Note that these are the first experimental results on the product of the length-lex and subset-bound with cardinality domains.

Social Golfer Problem The ll-domain uses the model and propagators given in [17, 14]. The sbc-domain uses the model in [20] and the $pairatmost1$ propagator in [18]. The ls-domain uses both models, with a channeling constraint to link the corresponding variables in the two models. The models use the same static labeling technique: variables are labeled in a week-wise fashion and, within each week, the variable with the largest domain is selected first (ties are broken by smaller group index first), and the choice consists in inserting the smallest element first and to exclude it on backtracking. BDD-SAT-VSIDS encodes the set domains with binary decision diagrams (BDDs) which are then integrated with a SAT solver for clauses generation and no-good learning. The SAT formulation employs a VSIDS search [21] that features a variable selection heuristics and

	ll-domain		**ls-domain**		BDD-SAT-VSIDS
g,s,w	Time	Fails	Time	Fails	Time
48 Easy	1.52	181	**1.38**	**137**	6.14
4,3,5	0.05	69	**0.04**	**64**	0.46
5,3,6	3.93	2728	2.47	**1991**	**0.8**
5,3,7	17.06	7650	12.1	**6274**	**6.32**
5,4,5	0.35	218	**0.19**	**147**	0.98
5,5,4	0.17	89	0.14	**77**	**0.05**
5,5,5	0.25	87	0.22	**75**	**0.07**
5,5,6	0.21	60	0.17	**47**	**0.15**
5,5,7	**0.01**	**1**	**0.01**	**1**	3.28
6,5,5	51.52	17197	21.61	**9948**	**15.24**
7,6,2	0.04	**14**	**0.03**	**14**	1.44
10,3,9	1.15	82	**0.77**	**5**	16.66
10,3,10	1.52	82	**1.08**	**12**	110.8
Total	77.77	28458	**40.2**	**18792**	162.39

Table 1. The Behavior of Set Domains on the Social Golfer Problem.

restarts. BDD-SAT-VSIDS runs on a 3.00GHz Core 2 Duo with 2 GB of memory. Table 1 gives a comprehensive comparison between these approaches on the social golfer problem. The ls-domain explores the smallest search tree and is the most robust overall. It is about twice as fast as the ll-domain and 4 times faster than BDD-SAT-VSIDS, which provides state-of-the-art results. The ll-model is also more than twice as fast as the BDD-SAT-VSIDS.

The Steiner Triple System The steiner triple system with n elements is a special class of balanced incomplete block design by setting $v = n, b = n(n-1)/6, r = (n-1)/2, k = 3, \lambda = 1$ [22]. It can be modeled using a dual set of set variables in which a primal variable X_i $(1 \leq i \leq v)$ corresponds to a point and a dual variable Y_j $(1 \leq j \leq b)$ corresponds to a block. In steiner triple system, every pair of points have exactly one common element (i.e., $|X_i \cap X_j| = 1, \forall i \neq j$). Complete filtering algorithm for the ll-domain is given in [17]. For the sbc-domain, the problem can be expressed as a conjunction of $pairatmost1$ [18] and $nonEmptyIntersection$ [19]. The ls-domain uses both model with channeling constraints to link them. All models apply a static smallest-index first labeling technique in the primal variable. For the BDD-SAT formulation, two search strategies are given: a static labeling and the VSIDS search discussed above.

Table 2 compares different approaches on Steiner Triple System. In the BDD-SAT formulation, $\times$ indicates a timeout of 900 seconds, and unreported instances are left as blanks. The ls-domain is the fastest in average and on most instances. It may bring significant benefits compared to the ll-domain and, especially over BDD-SAT. In particular, the BDD-SAT formulation appears much less robust. Once again, the experimental results indicate that the rich ls-domain provide state-of-the-art performance, despite its potentially high complexity.

	ll-domain		**ls-domain**		BDD-SAT-static	BDD-SAT-VSIDS
n	Time	Fails	Time	Fails	Time	Time
7	**0.01**	**0**	**0.01**	**0**	**0.01**	0.03
9	**0.01**	**1**	**0.01**	**1**	0.02	0.02
13	**0.01**	**1**	0.02	51	0.06	0.02
15	**0.03**	**1**	0.04	73	0.07	0.32
19	**0.12**	6	0.17	140	0.37	0.07
21	**0.29**	**30**	0.35	192	0.82	39.19
25	**4.84**	943	5.63	**912**	7.1	×
27	14.01	**650**	**12.61**	1826	12.88	229.59
31	2.86	**0**	**2.03**	**0**	5.38	×
33	35.94	**627**	24.32	1714	443.07	**19.3**
37	×	×	**1345.34**	**16178**		
39	1001.74	12040	**505.57**	**5248**		

Table 2. The Behavior of Set Domains on Steiner Triple System.

Algorithm 1 $bc_{ls}\langle atmost1(\{s_1,..,s_m\})\rangle(X_{ls} = ls\langle l,u,n,r,p\rangle)$

1: $\mathcal{S} \leftarrow \{s \in X_{ls} \mid \bigwedge_{1\leq i\leq m} |s \cap s_i| \leq 1\}$
2: $l', u' \leftarrow \min_{\preceq} \mathcal{S}, \max_{\preceq} \mathcal{S}$
3: $r', p' \leftarrow \bigcap_{s\in\mathcal{S}} s, \bigcup_{s\in\mathcal{S}} s$
4: **return** $ls\langle l', u', n, r', p'\rangle$

5 Exponential Filtering for Intersection Constraints

The previous sections reported intriguing theoretical and experimental results. The theory indicated that constraint propagation of even simple constraints may take exponential time in the worst case for the length-lex domain, while the experimental results clearly showed that the product of length-lex and subset-bound domains led to the best and most robust performance on some standard benchmarks. In this section, we reconsider the intractable unary intersection constraint. Instead of decomposing them into simpler unary constraints, we propose simple exponential algorithms for enforcing bound consistency on the ll- and ls-domains. Our motivation is twofold:

1. An exponential filtering algorithm enables us to move the potentially exponential behavior from the rather agnostic constraint propagation algorithm into the constraint itself where the constraint semantics can be exploited.
2. The stronger filtering further increases the pruning of the search and may lead to additional domain reduction through constraint propagation of other constraints, an observation already pointed out in [1].

Algorithm 1 implements $bc_{ls}\langle atmost1(\{s_1,..,s_m\},X)\rangle$ and is self-explanatory. The set $\mathcal{S}$ maintains a logical enumeration of all possible solutions. All four bounds of the ls-domain are determined according to the ls-bound-consistency

Algorithm 2 $bc_{ll}\langle atmost1(\{s_1, .., s_m\})\rangle(X_{ls} = ls\langle l, u, n, r, p\rangle)$

1: $l' \leftarrow \min_{\preceq} \{s \in X_{ls} \mid \bigwedge_{1 \leq i \leq m} |s \cap s_i| \leq 1\}$
2: $u' \leftarrow \max_{\preceq} \{s \in X_{ls} \mid \bigwedge_{1 \leq i \leq m} |s \cap s_i| \leq 1\}$
3: **return** $ls\langle l', u', n, r, p\rangle$

definition. Corollary 3 implies that there are no fixed-parameter tractable algorithm for Algorithm 1 since $hs_{ll}\langle\mathcal{C}\rangle$ is a special case for $bc_{ls}\langle\mathcal{C}\rangle$.

Theorem 4. *Algorithm 1 runs in time $O(n^c mc)$ where $c = |u|$.*

Proof. X_{ls} contains at most $O(n^c)$ sets. Each set takes $O(mc)$ time to verify if it satisfies the constraint. □

Sometimes enumerating all possible solutions is not cost-effective and hence we also consider an exponential filtering algorithm (Algorithm 2) for the length-lex bounds only. Obviously, lines 1–2 do not compute the set of solutions explicitly but only searches for the smallest and largest solution in the length-lex ordering. Algorithm 2 has the same worst case time complexity as Algorithm 1, since its feasibility routine is $W[1]$-hard, but it may be significantly faster in practice. The same principles can be applied to other unary intersection constraints.

6 Experimental Results

Social Golfer Problem Since the model uses a vanilla week-wise labeling strategy, when we label week w, all variables $X_{w',g'}$ in earlier weeks ($\forall w' < w$) are bounded. As a consequence, we can set $s_i = X_{w',g'}$ and introduce the $atmost1$ constraint. We apply the following rule for all variables $X_{w,g}$:

$$\frac{\forall w' < w, g' : \;\; X_{w',g'} \text{ is bounded}}{\forall w' < w, g' : \; |X_{w,g} \cap X_{w',g'}| \leq 1 \;\longmapsto atmost1(\{X_{w',g'} \mid w' < w\}, X_{w,g})}$$

Table 3 reports the experimental results on the $atmost1$ propagator using instances in Table 1 and 40 larger and harder instances. The ls-domain model with $bc_{ls}\langle atmost1\rangle$ is the only model to solve all instances within the time limit. It is the fastest model for all but one instance. On the traditional instances, the model is more than 6 times faster than the BDD-SAT-VSIDS approach. It is important to note that the search procedure is static and completely uninformed: All the reasoning is taking place in the propagation. The results also show the complementary between the exponential propagator and the richer domain, since the ll-domain, even with the exponential propagator for the length-lex component, is not as fast and robust.

The exponential propagators proposed in this paper have two roles: They increase the amount of filtering and they accelerate the convergence of the fixpoint

	BDD-SAT -VSIDS	ls-domain		ll-domain + $bc_{ll}\langle atmost1\rangle$		ls-domain + $bc_{ls}\langle atmost1\rangle$	
g,s,w	Time	Time	Fails	Time	Fails	Time	Fails
58 Easy	140.83	6.49	2570	9.67	3649	**5.98**	**1808**
5,3,7	**6.32**	12.1	6274	17.82	7660	10.31	**5373**
6,5,5	15.24	21.61	9948	36.26	17245	**8.42**	**4841**
Sub-total	162.39	40.2	18792	63.75	28554	**24.71**	**12022**
5,4,6		196.2	104570	258.6	130658	**102.5**	**56174**
6,5,6		232.8	65734	352.2	107362	**79.06**	**29578**
6,6,4		×	×	×	×	**1775**	**890645**
7,3,8		0.2	12	0.21	13	**0.17**	**8**
7,4,6		**0.14**	**5**	0.15	6	**0.14**	20
7,5,5		0.38	96	0.57	228	**0.15**	**11**
7,6,4		0.61	196	0.66	312	**0.24**	**58**
7,7,4		272.8	44954	2.82	895	**1.6**	**333**
8,3,10		478.6	59610	1435	222609	**166.2**	**26282**
8,4,7		2.49	416	14.57	5125	**0.62**	**111**
8,5,6		11.42	2448	29.16	10166	**1.07**	**333**
8,6,5		37.34	7080	43.06	15740	**2.64**	**1081**
8,7,4		6.68	1592	10.42	4160	**1.18**	**413**
9,3,11		49.96	4254	191.7	26745	**16.46**	**1990**
9,4,8		29.75	4031	73.29	16732	**3.26**	**496**
9,5,7		414.5	46835	×	×	**16.08**	**3462**
9,6,6		×	×	×	×	**55.56**	**17332**
9,7,5		×	×	×	×	**20.9**	**6923**
9,8,4		657	97659	1164	282668	**37.21**	**14609**
10,3,12		12.48	909	×	×	**2.33**	**61**
10,4,9		20.34	1926	363.7	54956	**2.56**	**137**
10,5,7		4.21	572	79.36	14340	**1.59**	**64**
10,6,6		17.33	2386	13.39	2125	**3.36**	**454**
10,7,5		11.54	1673	115.4	21687	**3.18**	**286**
10,8,4		39.98	**3457**	85.17	12595	**38.9**	3560
11,3,13		8.19	310	135.4	14005	**3.42**	**18**
11,4,10		11.68	606	849.3	98803	**2.18**	**12**
11,5,8		628.3	62145	76.52	10199	**15.09**	**2331**
11,6,7		1497	115092	×	×	**18.17**	**2687**
11,7,5		5.15	100	3.66	144	**3.21**	**81**
11,8,3		427.5	**26379**	274.9	36739	**187.8**	30144
11,9,3		338	**38996**	465.5	60785	**283.6**	43524
11,10,3		24.09	**2545**	43.35	5454	**22.09**	3045
12,3,14		12.38	382	864.79	77133	**8.85**	**217**
12,4,11		×	×	×	×	**84.9**	**5459**
12,5,9		×	×	×	×	**70.05**	**8639**
12,6,7		132.7	19374	76.29	7403	**6.25**	**114**
12,7,6		21.91	622	×	×	**20.24**	**424**
12,8,4		**39.45**	**2502**	49.44	3019	41.18	2630

Table 3. The Benefits of the *atmost*1 Constraint on the Social Golfer Problem.

	ls-domain		ls-domain + $bc_{ll}\langle atmost1\rangle$		ls-domain + $bc_{sb}\langle atmost1\rangle$		ls-domain + $bc_{ls}\langle atmost1\rangle$	
(g,s,w)	Time	Fails	Time	Fails	Time	Fails	Time	Fails
5,3,7	12.1	6274	11.73	6250	10.55	**5376**	**10.31**	**5373**
5,4,6	196.2	104570	169.4	102385	112	56260	**102.5**	**56174**
6,5,5	21.61	9948	15.88	9536	10.52	4849	**8.42**	**4841**
6,5,6	232.8	65734	169.5	63113	97.21	29639	**79.06**	**29578**
6,6,4	×	×	**1687**	958106	×	×	1775	890645
9,3,11	49.96	4254	31.67	4248	20.06	**1989**	**16.46**	1990
9,4,8	29.75	4031	15.17	3994	5.09	**495**	**3.26**	496
9,5,7	414.5	46835	156.1	44247	31.77	**3460**	**16.08**	3462
9,6,6	×	×	1718	566144	121.6	**17332**	**55.56**	**17332**
9,7,5	×	×	×	×	59.31	**6923**	**20.9**	**6923**
9,8,4	657	97659	172.5	72688	107.6	**14609**	**37.21**	**14609**
12,3,14	12.38	382	11.75	382	9.33	**217**	**8.85**	**217**
12,4,11	×	×	1018	99989	128.3	**5457**	**84.9**	5459
12,5,9	×	×	×	×	97.76	**8639**	**70.05**	**8639**
12,6,7	132.7	19374	126.1	19370	13.08	**114**	**6.25**	**114**
12,7,6	21.91	622	19.77	626	50	**424**	**20.24**	**424**
12,8,4	39.45	**2502**	46.1	2631	52.76	2630	**41.18**	2630

Table 4. An Analysis of the Benefits of the *Atmost*1 Constraint.

algorithm. Table 4 shows that the effects are cumulative on the social golfer problem. The results are obtained on two difficult instances from the standard benchmarks in Table 1 (as the remaining ones are too easy) as well as from larger and more difficult instances. The results compare the ls-domain with three versions of the exponential propagator: bc_{ll} which only updates the length-lex bounds, bc_{sb} which only updates the required and possible, and the complete propagator bc_{ls}. $bc_{ls}\langle atmost1\rangle$ and $bc_{sb}\langle atmost1\rangle$ should produce the same search tree, while $bc_{ll}\langle atmost1\rangle$ and ls-domain should also explore the same but larger tree since these models do not have extra propagation on the required and possible sets. [3] Therefore the comparison between $bc_{ls}\langle atmost1\rangle$ and $bc_{sb}\langle atmost1\rangle$ on the one hand and $bc_{ll}\langle atmost1\rangle$ and ls-domain on the other hand measures how much the $atmost1$ propagator speeds up the convergence of the fixpoint algorithm. The comparison between $bc_{ls}\langle atmost1\rangle$ and $bc_{ll}\langle atmost1\rangle$ measures the benefits from the additional pruning obtained by propagating both bounds simultaneously.

Observe first that $bc_{sb}\langle atmost1\rangle$ can be significantly slower than $bc_{ls}\langle atmost1\rangle$. Instance (9,8,4) is particularly interesting in that regard. $bc_{ls}\langle atmost1\rangle$ takes about 37 seconds, while $bc_{sb}\langle atmost1\rangle$ completes in about 107 seconds. So, on

[3] There are some differences between the number of failures in our implementation since the fixpoint algorithm can halt prematurely, i.e., when the number of iterations exceeds a threshold based on the number of constraints. Similar techniques are used for continuous constraint propagation.

this instance, $bc_{ls}\langle atmost1\rangle$ speeds up the propagation by a factor close to 3 by using an exponential propagator. However, the results for $bc_{ll}\langle atmost1\rangle$ show the significant benefits of additional propagation coming from the product of the length-lex and subset-bound domains. Since both of them use the exponential propagator for the length-lex component, the benefits come from the additional filtering, not the speed of the fixpoint algorithm. The experimental results show significant improvements in efficiency in favor of the richer domain. For instance, $bc_{sb}\langle atmost1\rangle$ terminates after 20 and 70 seconds on instances $(9, 7, 5)$ and $(12, 5, 9)$, while $bc_{ll}\langle atmost1\rangle$ does not terminate after 1,800 seconds.

Steiner Triple System The model uses a static smallest index variable first labeling strategy. When we label variable X_i, all lower indexed variables $X_{i'}$ $(\forall i' < i)$ are bounded. We set $s_{i'} = X_{i'}$ and introduce the $exact1$ constraint.

$$\frac{\forall i' < i: \quad X_{i'} \text{ is bounded}}{\forall i' < i,\ |X_i \cap X_{i'}| = 1 \ \longmapsto exact1(\{X_{i'} \mid i' < i\}, X_i)}$$

Table 5 reports the experiment results on the $exact1$ propagator using the instances in Table 2. Here the ls-domain with the length-lex bound propagator ($bc_{ll}\langle exact1\rangle$) gives the best results and improves the state-of-the-art considerably. It solves two more instances that are unsolvable by the original model and is several orders of magnitude faster than the original model and the BDD-SAT formulations. *What is particularly remarkable is that the model has almost no failure on these instances and that the only benefit of the exponential propagator is to speed up the computation of the fixpoint algorithm.* The complete filtering algorithm ($bc_{ls}\langle exact1\rangle$) is much slower since it must explore a considerable number of sets. The additional pruning that it brings is not compensated by the increased computational costs. What this benchmarks shows is that speeding up the fixpoint algorithm by exponential propagators may produce substantial improvements in efficiency.

7 Related Work

This section briefly reviews some related work on exponential propagation and propagators. Perhaps the closest related work is the work on box consistency in the Numerica system [23]. The key idea of box consistency was to avoid the decomposition of a complex constraints into elementary ternary constraints. By enforcing box consistency on the original constraint, these systems improve the pruning, addresses the so-called dependency effect of interval propagation, and tackle the fact that the fixpoint algorithm can take a long time to converge. Box consistency was enforced by a potentially exponential algorithm. The Newton and Numerica systems also include conditions to terminate the fixpoint algorithm prematurely when the propagation was not reducing the search space enough. Lebbah and Lhomme [12] considered the use of extrapolation methods to speed up the convergence of filtering algorithms for continuous CSPs, also dramatically the efficiency on these problems. These techniques could potentially

	ls-domain		ls-domain + $bc_{ll}\langle exact1\rangle$		ls-domain + $bc_{ls}\langle exact1\rangle$	
n	Time	Fails	Time	Fails	Time	Fails
7	**0.01**	**0**	**0.01**	**0**	**0.01**	**0**
9	**0.01**	**1**	**0.01**	**1**	**0.01**	**1**
13	0.02	51	**0.01**	**1**	0.02	**1**
15	**0.04**	73	**0.04**	**1**	**0.04**	**1**
19	0.17	140	**0.15**	**0**	0.19	**0**
21	0.35	192	**0.26**	**0**	0.40	**0**
25	5.63	912	**0.55**	**0**	2.66	**0**
27	12.61	1826	**0.89**	**0**	4.74	**0**
31	**2.03**	**0**	2.22	**0**	2.95	**0**
33	24.32	1714	**4.33**	1	19.47	**0**
37	1345.76	16178	**24.24**	**0**	1443.32	**0**
39	505.57	5248	**21.92**	**0**	1307.45	**0**
43	×	×	**403.20**	**0**	×	×
45	×	×	**1344.79**	**0**	×	×

Table 5. The Benefits of the *exact*1 Constraint on the Steiner Triple System.

be applied to set domains as well, but this paper took another, simpler, route: Using exponential propagators that have a more global view of the problem at hand. Also closely related is the work of Bessiere and Régin on solving CSPs on the fly. They recognize that, on certain applications, the pruning offered by the solver was not strong enough. They isolated a global constraint (i.e., the sum of n variables taking different values) for which they did not design a specific propagator. Instead, they use the CP solver recursively and solved CSPs on the fly to enforce arc consistency. Once again, the result is to move some of the exponential behavior from the search to the constraint propagation. Note also that several pseudo-polynomial algorithms have also been proposed in the past, including the well-known filtering algorithm for knapsack constraints [24].

8 Conclusion

Most research in constraint programming focuses on designing polynomial-time filtering algorithms. This paper explored, for set CSPs, the idea of shifting some of the exponential behavior from the search component to the filtering component, and from the constraint-propagation algorithm to the propagators. It showed that the product of the length-lex and subset-bound domains improves state-of-the-art results significantly despite a potential exponential propagation algorithm. More importantly, it presented exponential-time propagators for intractable unary intersection constraints and demonstrated that they bring considerable performance improvement by speeding up constraint propagation and increasing filtering. They indicate that it may sometimes be beneficial to embrace complexity in the filtering component and exploit the constraint semantics and locality, instead of relying on rather agnostic search and constraint propagation algorithms.

References

1. Bessière, C., Régin, J.C.: Enforcing arc consistency on global constraints by solving subproblems on the fly. In Jaffar, J., ed.: CP. Volume 1713 of Lecture Notes in Computer Science., Springer (1999) 103–117
2. Puget, J.F.: Pecos a high level constraint programming language. In: Proc. of Spicis. (1992)
3. Gervet, C.: Interval propagation to reason about sets: Definition and implementation of a practical language. Constraints **1**(3) (1997) 191–244
4. Sadler, A., Gervet, C.: Global reasoning on sets. In: In Proceedings of Workshop on Modelling and Problem Formulation (FORMUL01). held alongside CP-01. (2001)
5. Azevedo, F.: Cardinal: A finite sets constraint solver. Constraints **12**(1) (2007) 93–129
6. Bessière, C., Hebrard, E., Hnich, B., Walsh, T.: The complexity of global constraints. In McGuinness, D.L., Ferguson, G., eds.: AAAI. (2004) 112–117
7. Peter Hawkins, V.L., Stuckey, P.J.: Solving set constraint satisfaction problems using robdds. Journal of Artificial Intelligence Research **24** (2005) 109–156
8. Gervet, C., Van Hentenryck, P.: Length-lex ordering for set csps. In: AAAI. (2006)
9. Sellmann, M.: On decomposing knapsack constraints for length-lex bounds consistency. In: CP'09. (2009) 762–770
10. Sadler, A., Gervet, C.: Hybrid set domains to strengthen constraint propagation and reduce symmetries. In: CP'04. (2004) 604–618
11. Malitsky, Y., Sellmann, M., Van Hoeve, W.: Length-Lex Bounds Consistency for Knapsack Constraints. In: CP'08. (2008) 266–281
12. Lebbah, Y., Lhomme, O.: Accelerating filtering techniques for numeric csps. Artif. Intell. **139**(1) (2002) 109–132
13. Mercier, L., Van Hentenryck, P.: Edge finding for cumulative scheduling. INFORMS Journal on Computing **20**(1) (2008) 143–153
14. Yip, J., Van Hentenryck, P.: Evaluation of length-lex set variables. In Gent, I.P., ed.: CP. Volume 5732 of LNCS, Springer (2009) 817–832
15. Downey, R.G., Fellows, M.R.: Fixed-parameter tractability and completeness ii: On completeness for w[1]. Theor. Comput. Sci. **141**(1&2) (1995) 109–131
16. Gange, G., Lagoon, V., Stuckey, P.: Fast set bounds propagation using a bdd-sat hybrid. to appear in JAIR (2010)
17. Van Hentenryck, P., Yip, J., Gervet, C., Dooms, G.: Bound consistency for binary length-lex set constraints. In Fox, D., Gomes, C.P., eds.: AAAI. (2008) 375–380
18. van Hoeve, W.J., Sabharwal, A.: Filtering atmost1 on pairs of set variables. In Perron, L., Trick, M.A., eds.: CPAIOR. Volume 5015., Springer (2008) 382–386
19. Yip, J., Van Hentenryck, P., Gervet, C.: Boosting set constraint propagation for network design. CPAIOR (2010)
20. Barnier, N., Brisset, P.: Solving the kirkmans schoolgirl problem in a few seconds. In: CP-2002, Springer-Verlag (2002) 477–491
21. Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., Malik, S.: Chaff: Engineering an efficient sat solver. (2001) 530–535
22. Colbourn, C.J., (Eds.), J.H.D., Dinitz, J.H., Ii, L.C., Jajcay, R., Magliveras, S.S.: The crc handbook of combinatorial designs (1995)
23. Van Hentenryck, P., Michel, L., Deville, Y.: Numerica: a Modeling Language for Global Optimization. The MIT Press, Cambridge, Mass. (1997)
24. Trick, M.A.: A dynamic programming approach for consistency and propagation for knapsack constraints. Annals OR **118**(1-4) (2003) 73–84