Anomalies in SMT Solving: Difficulties in Modelling Combinatorial Problems

Alan M. Frisch[1] and Miquel Palahí[2]

[1] University of York, UK
[2] Universitat de Girona, Spain

Abstract Much research on modelling combinatorial problems for a solver compares alternative models of a problem in an attempt to understand the characteristics of good models. The aim is to discover principles and heuristics that in the future would guide someone to develop good models or to select the more effective model from a set of alternatives. For many years this methodology has been moderately successful in studying modelling for SAT solvers and for finite-domain constraint solvers.
Our attempts to apply this methodology to SMT solving have been hindered by the apparently erratic behaviour of SMT solvers. This paper presents four of the more extreme anomalies that we have encountered in our work. Of course we label these phenomena as anomalies because we we have no explanation for them and they run counter to our intuitions. We bring these anomalies to light in an attempt to motivate the research community to try to develop a better understanding of SMT solving and modelling.

1 Introduction

SAT modulo theory (SMT) solvers combine the powerful technology of modern SAT solvers with specialised reasoning that supports particular theories. SMT was originally developed for verification problems and has proven to be highly effective in that domain. Some recent effort has been invested in solving combinatorial decision and optimisation problems with SMT, particularly exploiting specialised reasoning over linear integer arithmetic.

The most significant study [1] of using SMT for combinatorial problems developed an automatic system to translate constraint problem instances expressed in FlatZinc to SMT. Over a suite of 294 instances of 32 problems the solve times of the Yices 2 SMT solver—predominantly using QF_LIA, the theory of quantifier free linear integer arithmetic–were competitive with those of leading finite-domain constraint solvers.

The strong performance of the SMT solver is both striking and highly promising when one considers the context in which the comparison was made. Firstly, the source language of the problem instances, FlatZinc, is designed primarily with finite-domain constraint solvers in mind; this could disadvantage the SMT

solver. Secondly, the efforts of the Girona group focused mostly, though not exclusively, on exploring the breadth of SMT's capabilities rather than on how *best* to encode combinatorial problems in SMT.

The line of research we have been pursuing is complementary to that of the Girona group. We aim to better understand how best to model combinatorial problems in SMT, free from the language of FlatZinc and particular FlatZinc problem encodings. Our work towards this goal has used two two methodologies that have proven effective for both finite-domain constraint solvers and SAT solvers. One is to discover how best to encode particular problems and from this draw generalisations that could be applied to modelling other problems. The other approach is to consider how best to encode particular components that commonly arise in combinatorial problems, such as finite domains or certain constraints. In both cases the goal is form general principles or patterns that can be used in the future to build effective encodings of other problems.

Disappointingly and surprisingly our attempts to identify patterns and generalisations were hampered by erratic behaviour of SMT solvers. An encoding that worked well for one solver would be poor for another. Changes to encodings that practitioners of finite-domain constraint programming would expect to be improve performance, such as tightening bounds on the objective function, sometimes proved to hamper performance.

This paper reports on the anomalous behaviour of SMT solvers using QF_LIA that has impeded our progress towards constructing an understanding of how best to model combinatorial problems for SMT solvers. The work reported here is solely concerned with SMT using the QF_LIA theory and we refer to this combination as SMT(QF_LIA). We investigate four SMT(QF_LIA) solvers and use all with their default settings.

Problem instances and data associated with this research can be found at http://www.cs.york.ac.uk/aig/constraints/SMT/.

2 Anomalies in Representing Finite Domains

The first two anomalies arise in representing finite integer domains and considering two kinds of propagation.

Throughout we consider the representation of a variable x with domain $D = \{d_1, \ldots, d_n\}$ where $d_1 \leq d_2 \leq \cdots \leq d_n$. We use $\overline{D}$ to denote $\{d' | d_1 < d' < d_n and d' \notin D\}$.

To simplify the presentation we use the following schemas:

- $ALO(x) \stackrel{\text{def}}{=} \bigvee_{d \in D} x = d.$
- $BOUND(x) \stackrel{\text{def}}{=} (d_1 \leq x \leq d_n).$
- $NEG(x) \stackrel{\text{def}}{=} \bigwedge_{d \in \overline{D}} \neg(x = d).$

In all the experiments of Anomalies 1 and 2 we have used the following four SMT(QF_LIA) solvers: Yices 1.0.35 [6], Yices 2.1.0, Z3-4.1 [5] and MathSat 5.1.10 [2]. The input formulas for the first three solvers are written using

SMT-Lib 1.2 and using SMT-Lib 2 for the fourth. The experiments were run on a cluster of Intel® Xeon™CPU@3.1GHz machines, with 8GB of RAM, under 64-bit CentOS release 6.3 (kernel 2.6.32).

2.1 Anomaly 1

Anomaly 1 arises in a test to measure how fast different finite domain representations are in recognising unsatisfiability when all of the domain values are eliminated.

Here we consider the domain of variable x to be the contiguous set of values $1..n$. The domain values are all eliminated by asserting the following formula:

$$UNSAT\_ALL(x) \stackrel{\text{def}}{=} \bigwedge_{d \in D} \neg(x = d)$$

We conjoin this with each of three domain representations resulting in three test cases

1. $ALO(x) \wedge UNSAT\_ALL(x)$
2. $BOUND(x) \wedge UNSAT\_ALL(x)$
3. $BOUND(x) \wedge ALO(x) \wedge UNSAT\_ALL(x)$

We measured the solve time of each of these formulas on each of the four SMT(QF_LIA) solvers. Each of Figures 1–4 shows the run time as a function of domain size, n, of one solver on the three domain encodings. The first three of these figures show that each of Yices 1, Yices 2 and MatSat 5 perform similarly on the three encodings. Indeed, in each case two of the curves overlay each other almost perfectly. MathSat 5 is remarkable in that the solve time is almost independent of both domain size and encoding.

The anomaly occurs with Z3; as domain size increases the bounds-only encoding performs far worse than the other two encodings, which behave almost identically. Notice that the scale in Fig. 4 goes over 100 seconds, whereas the scales in the other plots go to only 10 seconds. More remarkably, with the bounds-only representation Z3 performs search. For example, with domain size 10000 it reports having made 5843 decisions and 5460 conflicts, while in the ALO representations does not report any information about decisions or conflicts.

In contrast, the other solvers don't report having made any decision as expected, but they report different number of conflicts. On all problem instances Yices 1 reports 1 conflict, Yices 2 does not report any statistic when the instance is unsatisfiable and MathSat 5 reports 1 conflict with the bounds-only representation and 0 conflicts in the ALO representations. Finally, MathSat 5 is the only solver reporting calls to theory solvers, calling 5001 times the linear arithmetic theory with the domain size 10000. A summary of the reported conflicts and calls to the linear arithmetic theory solver by the SMT solvers can be found in Table 1.

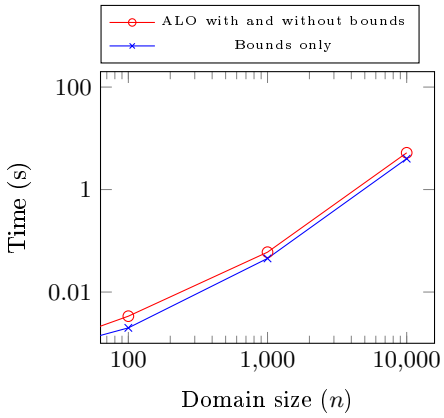

Figure 1. Anomaly 1: Yices 1

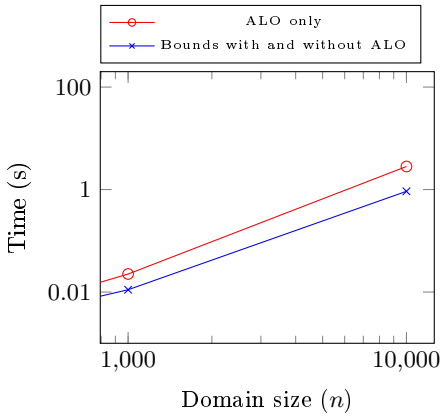

Figure 2. Anomaly 1: Yices 2

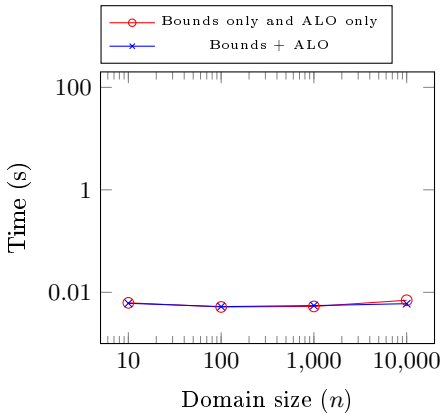

Figure 3. Anomaly 1: MathSat 5

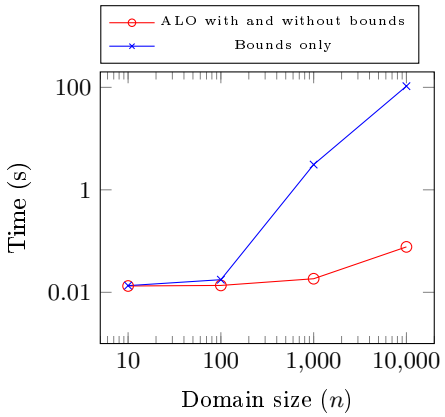

Figure 4. Anomaly 1: Z3

2.2 Anomaly 2

Anomaly 2 arises in a test to measure how fast different domain representations are in recognising that a variable has a particular value because all but one of it's domain values are eliminated.

Here we consider the domain of variable x to be the first n odd natural numbers, $\{1, 3, 5, \ldots, 2n-1\}$. All but one of the domain values are eliminated by asserting the following formula:

$$ONLY1(x, v) \stackrel{\text{def}}{=} \bigwedge_{d \in D \setminus \{v\}} \neg(x = d)$$

We have evaluated the performance of four solvers in determining the satisfiability of $ONLY1(x, d_n)$ when conjoined with each of four domain represent-

	Bounds	ALO	Bounds + ALO
Yices 1	1	1	1
Yices 2	n/a	n/a	n/a
Z3	5460	n/a	n/a
MathSat 5	1 (5001)	0 (0)	0 (0)

Table 1. Anomaly 1: Number of conflicts and number of calls to the linear arithmetic theory solver (between parenthesis) for instance with domain size 10000 for each SMT solver. *n/a* means that the solver has not reported any information.

ations. Anomaly 2 arises in the conjunction with one particular domain representation

$$BOUND(x) \wedge NEG(s) \wedge ONLY1(x, d_n)$$

This formula has been solved with each of the four SMT(QF_LIA) solvers. The resulting solve times as a function of domain size, n, are shown in Figure 5. Here we see that Yices 2 and MathSat 5 scale very well with increasing n. In contrast, the solve time of Yices 1 increases rapidly with increasing n and if n is 13,000 or more the solver encounters a memory overflow.

The main anomaly here is that the solve time of Z3 does not increase monotonically with n. Furthermore, Z3 performs search in solving this simple problem and the metrics that quantify this search also do not increase monotonically. As shown in Figure 6, their non-monotonic behaviour tracks that of the runtime measurement. In contrast, the three other solvers report a constant number of conflicts and decisions; on all instances Yices 1 and Yices 2 report 0 conflict and MathSat 5 reports 1 conflict. MathSat makes $2n + 3$ calls to the QF_LIA theory solver on instances with domain size n; the other solvers do not report this statistic.

3 Anomalies in representing the Pedigree Reconstruction Problem

The second two anomalies arise in representing the Pedigree Reconstruction Problem (PRP)[4], which involves identifying relatives amongst a group G of individuals from genetic marker data.

In particular, the goal here is to find the maximum likelihood pedigree. A pedigree for G assigns to each individual i in G a parent set, which takes one of three forms:

- j, k, where i, j and k are distinct members of G. This indicates that the parents of i are j and k.
- j, where i and j are distinct members of G. This indicates that j is a parent of i and the other parent of i is not in G.
- $\emptyset$, which indicates that neither parent of i is in G.

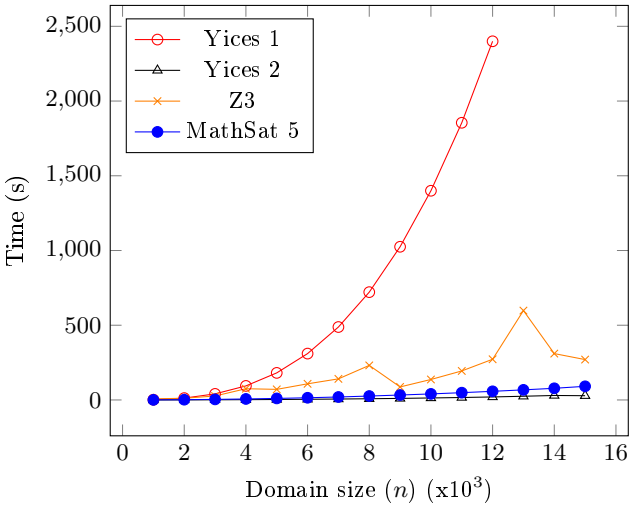

Figure 5. Anomaly 2

For each member $i \in G$, the genetic markers of the members of G determine a probability distribution over all possible parent sets of i. In typical problem instances most potential parent sets of i have probability zero.

The probability of a pedigree is the product of the probability that each $i \in G$ has the parent set assigned by the pedigree.

Every assignment of individuals to parent sets is not a valid pedigree. Sexual reproduction imposes two constraints:

acyclicity: No individual can be an ancestor of itself, and

gender consistency: The two parents of an individual have opposite gender. The genetic marker data does not identify the gender of the individuals, so gender consistency requires that a gender could be assigned to each individual so that the parents of every individual have opposite genders. As an example, an assignment in which a and b have a child, b and c have a child, and a and c have a child is not gender consistent.

Summing up, the pedigree reconstruction problem is: Given G a finite set of individuals and for each $i \in G$ a probability distribution over all possible parent sets of i, find a maximum likelihood assignment A of a parent set to each $i \in G$ such that A is acyclic and gender consistent.

Following Cussens [4] we simplify our model by assuming that we are given not the probability of each parent set but rather the log of that probability. This enables us to use a linear objective function: the sum of the logarithm of the probability of each assigned parentset. This value is to be maximised.

Table 2 shows the basic model written in SMT(QF_LIA) of the PRP. The model represents the individuals of G by the integers $1..n$. For each individual

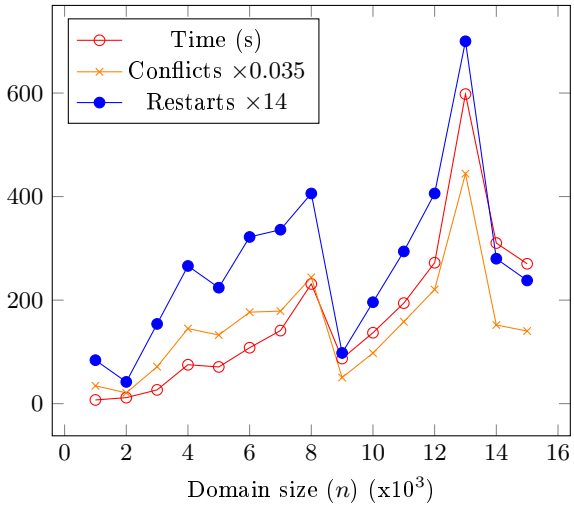

Figure 6. Z3 statistics in function of domain size (n). The number of decisions are not shown because the decisions $\times 0.033$ curve will look indistinguishable to the conflicts $\times 0.035$ curve.

i the instance specifies the probability distribution over the possible parent sets of i by three parameters.

- k_i is the number of possible parent sets of i that have non-zero probability
- $pps_i[1..k-i]$ is an array whose elements are the k_i parent sets i that have non-zero probability.
- $logLikelihood_i[1..k_i]$is an array such that element j is the log of the probability that i has parentset $pps_i[j]$.

For each individual i the model has a decision variable $parentset_i$ such that assigning it p indicates the decision to assign individual i the parentset $pps_i[p]$. The decision variable $globalValue$ is the objective value to be maximised. It is the sum of values of $localValue_i$ $(1 \leq i \leq n)$—see constraint (c3)— where $localValue_i$ is constrained by (c5) to be equal to $logLiklihood_i[parentset_i]$. Constraints (c1) and (c4) bound the $parentset_i$ and $globalValue$ variables, in effect giving each a finite domain. Constraint (c2) imposes an upper bound on the $localValue_i$ variables; the use of a lower bound is an issue discussed below.

Acyclicity is enforced by associating a generation number, the decision variable gen_i, with each individual i. The generation number of an individual is constrained to be one greater than the maximum generation number of its parents. This is stipulated in (c7) if i has two parents and in (c8) if i has one parent. Constraint (c6) bounds each gen_i, giving it a finite domain.

Finally, gender consistency is enforced by associating a gender, the boolean decision variable $female_i$ with each individual. The only constraint is that if an individual has two parents then those parents must have opposite gender. This is stipulated in the last conjunct of (c7).

Given

n:	positive integer	
k_i:	positive integer	$(1 \leq i \leq n)$
$pps_i[1..k_i]$:	set maxsize 2 drawn from $1..n$	$(1 \leq i \leq n)$
$logLikelihood_i[1..k_i]$:	positive integer	$(1 \leq i \leq n)$

Decision Variables

$parentset_i$: int	$(1 \leq i \leq n)$
gen_i: int	$(1 \leq i \leq n)$
$localValue_i$: int	$(1 \leq i \leq n)$
$globalValue$: int	
$female_i$:bool	$(1 \leq i \leq n)$

Constraints

(c1) $1 \leq parentset_i \wedge parentset_i \leq k_i \ (1 \leq i \leq n)$
(c2) $localValue_i \leq \max_{j \in 1..k_i} logLikelihood_i[j] \ (1 \leq i \leq n)$
(c3) $globalValue = \sum_{i=1}^{n} localValue_i$
(c4) $globalValue \geq 0 \wedge globalValue \leq \sum_{i=1}^{n} \max_{j \in 1..k_i} logLikelihood_i[j]$
(c5) not $(parentset_i = j) \vee localValue_i = logLikelihood_i[j] \ (1 \leq i \leq n, 1 \leq j \leq k_i)$
(c6) $0 \leq gen_i \wedge gen_i \leq n \ (1 \leq i \leq n)$
(c7) $(\text{not}(parentset_i = r) \vee gen_i - gen_p \geq 1) \wedge$
$(\text{not}(parentset_i = r) \vee gen_i - gen_{p'} \geq 1) \wedge$
$(\text{not}(parentset_i = r) \vee gen_i - gen_p = 1 \vee gen_i - gen_{p'} = 1) \wedge$
$(\text{not}(parentset_i = r) \vee not(female_p = female_{p'}))$
$(1 \leq i \leq n, 1 \leq j \leq k_i, pps_i[j] = \{p, p'\})$
(c8) not $(parentset_i = r) \vee gen_i - gen_p \geq 1 \ (1 \leq i \leq n, 1 \leq j \leq k_i, pps_i[j] = \{p\})$

Objective

maximize $globalValue$

Table 2. Basic SMT(QF_LIA) model of the pedigree reconstruction problem

The basic model imposes used upper and lower bounds on the variables *parentset* and *gen*, but only an upper bound on the variable $localValue$. This is because during our experiments we have detected surprising behaviour when we use a lower bound on that variable. Therefore, to study this behaviour, we have defined two variants of constraint (c2):

- (c2$^+$) when in addition to (c2) we also use the tighter lower bound:

$$localValue_i \geq \min_{j \in 1..k_i} logLikelihood_i[j]$$

- (c2^0) when in addition to $c2$ we also use a weaker lower bound:

$$localValue_i > 0$$

In some of the experiments we extended the model to use three additional constraints to represent the domain of the variables:
(c9) DomainALO(parentset): $\bigvee_{j \in 1..k_i} parentset_i = j \ (1 \leq i \leq n)$
(c10) DomainALO(gen): $\bigvee_{j \in 0..n} (gen_i = j) \ (1 \leq i \leq n)$
(c11) DomainALO(lc): $\bigvee_{j \in 1..k_i} localValue_i = logLikelihood_i[j]$

To simplify our study we replace the PRP optimisation problem with the corresponding decision problem. In particular, from each PRP instance we generate the hardest satisfiable instance and the hardest unsatisfiable instance. We precompute the global value, c^*, of the optimal solution and create two benchmark problem instances: a satisfiable instance in which the objective is replaced with the constraint $globalValue \geq c^*$ and an unsatisfiable instance in which the objective is replaced with the constraint $globalValue > c^*$.

Notice that in both the satisfiable and unsatisfiable instances the constraints (c10) and (c11) are implied. To fully appreciate Anomaly 4 it is important to bear in mind that because these are implied constraints they prune no solutions.

All of our experiments use the same suite of 100 PRP instances generated randomly by pedsim [3], configured to use genetically realistic random distributions. Such instances are known to be harder to solve than those generated using random distributions. All of our instances contain 46 individuals.

In all the experiments of Anomalies 3 and 4 we have used the same four SMT(QF_LIA) solvers used in Anomalies 1 and 2. But the experiments were run on a slightly different computer, Intel® Core™ i5 CPU@2.66GHz, with 3GB of RAM, under 32-bit openSUSE 11.2 (kernel 2.6.31).

3.1 Anomaly 3

Anomaly 3 arose through a surreptitious observation. Solve time can be reduced if we replace constraint (c3) with

$$(c3') \qquad globalValue2 = \sum_{i=1}^{n} localValue_i \wedge globalValue2 = globalValue$$

where $globalValue$ is a new integer variable.

This anomaly only appears in MathSat 5 and Yices 1 solvers when we are also using (c2$^+$) instead of (c2), and in the case of Yices 1 when we are also using ALO for the variables (constraints (c9), (c10) and (c11)). It appears only in the satisfiable instances and it does not appear in the unsatisfiable ones. Table 3 and Table 4 show the mean time and median time with and without the extra variable. In both cases the extra variable reduces the median time about 20%.

	Mean time (s)	Median time (s)
non ex.	5.1283	2.085
extra	4.0067	1.56

Table 3. Anomaly 3: MathSat 5 solves the 100 instances about 20% faster with an extra variable in the basic model with tighter lower bound.

	Mean time (s)	Median time (s)
non ex.	1.2569	1.005
extra	1.0584	0.82

Table 4. Anomaly 3: Yices 1 solves the 100 instances about 20% faster with an extra variable in the basic model with tighter lower bound and ALO.

One would expect that adding this extra variable would have only a trivial affect on solution time. However, the two scatter plots, Figure 7 and Figure 8, show that the effect of adding an extra variable can drastically increase or decrease the solving time of an instance. More precisely, in Figure 7 when adding the extra variable there is a variability on solving one instance from 52.9 times faster (decreasing the solving time from 35.92 to 0.68) to 32.4 times slower (increasing the solving time from 0.86 to 27.87). In Figure 8 this variability is smaller, ranging from 4.6 times faster (decreasing the solving time from 5.1 to 1.11) to 2.0 times slower (increasing the solving time from 1.51 to 3.02).

3.2 Anomaly 4

The last anomaly arises in test to measure in the basic model which way to represent the lower bound is the best for Z3 solver: without any lower bound (c2), with a tighter lower bound ($c2^+$) or with a lighter lower bound ($c2^0$). Again the first impression is that $c2^+$ has to be the best option, but in practise the best option is $c2$, that is not using any lower bound. This is shown at Table 5 where we can see the mean and median times of solving the 100 instances for the three experiments. The first two columns are the solving times for the satisfiable instances and the two last columns are the times for the unsatisfiable instances. We can see that solving the basic model with $c2$ is about 5 to 6 times faster than with $c2^+$ or $c2^0$ in the satisfiable case. A similar phenomenon happens with the unsatisfiable instances, where the basic model with $c2$ is about 3 to 4 times faster then the other two representations.

We want to note that this anomaly disappears if we add ALO to represent $localValue$ variable domain (adding constraint $c11$).

Finally, we present four scatter plots comparing $c2$ with $c2^+$ and $c2^0$ in Figure 9 and Figure 10 respectively for the satisfiable case and in Figure 11 and

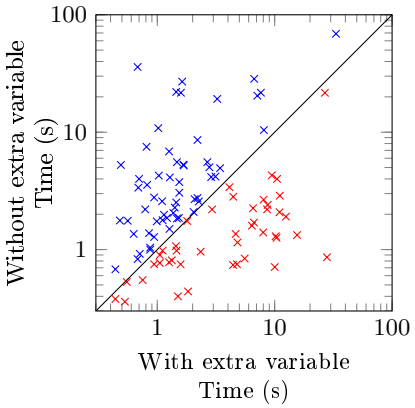

Figure 7. Anomaly 3: scatter-plot of MathSat 5 with and without extra variable in the basic model with tighter lower bound. 57 blue crosses and 43 red crosses.

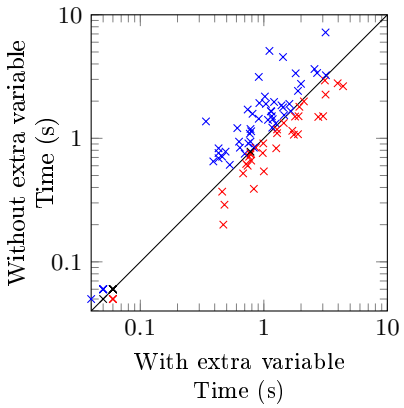

Figure 8. Anomaly 3: scatter-plot of Yices 1 with and without extra variable in the basic model with tighter lower bound and ALO. 53 blue crosses, 43 red crosses and 12 black crosses.

	Satisfiable		Unsatisfiable	
	Mean time (s)	Median time (s)	Mean time (s)	Median time (s)
$c2^+$	29.94	31.73	37.28	35.60
$c2^0$	35.43	34.64	41.47	38.76
$c2$	7.06	5.59	12.61	9.68

Table 5. Anomaly 4: Z3 solves the satisfiable instances about 5 to 6 times faster in the basic model without any lower bound compared to using any lower bound, and about 3 to 4 times faster for the unsatisfiable instances.

Figure 12 respectively for the unsatisfiable case. What is interesting of these plots is that in all the cases $c2$ is better in all the instances (specially for the unsatisfiable case). More precisely, $c2$ is better than $c2^+$ between 1.18 and 18.90 times in the satisfiable case and between 1.38 and 9.46 times better in the unsatisfiable case, and $c2$ is better than $c2^0$ between 1.17 and 22.56 times in the satisfiable case and between 1.54 and 9.71 times better in the unsatisfiable case.

4 Conclusions

The behaviours of the SMT(QF_LIA) solvers presented in this paper are anomalous in that they contradict our expectations, which have been formed from our experience with modelling problems for CP solvers. In Anomaly 1 we see that a simple, obvious representation of finite domains is problematic for one SMT(QF_LIA) solver, though not for three others.

Anomaly 2 arises in considering parameterised problem instances in which solving larger instances requires a superset of the reasoning involved in solving

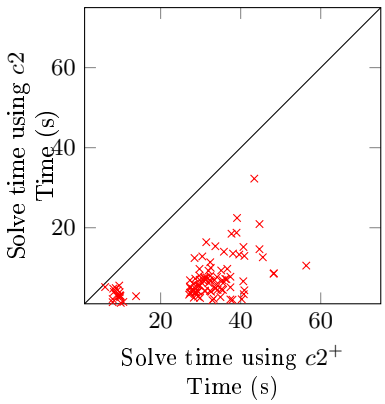

Figure 9. Anomaly 4: scatter-plot of Z3 with tighter lower bound ($c2^+$) and without lower bound ($c2$) in the basic model for the satisfiable case.

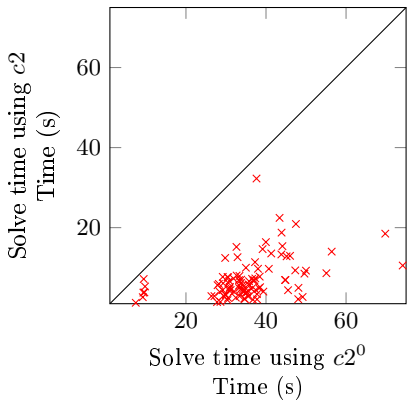

Figure 10. Anomaly 4: scatter-plot of Z3 with lower bound greater than 0 ($c2^0$) and without lower bound ($c2$) in the basic model for the satisfiable case.

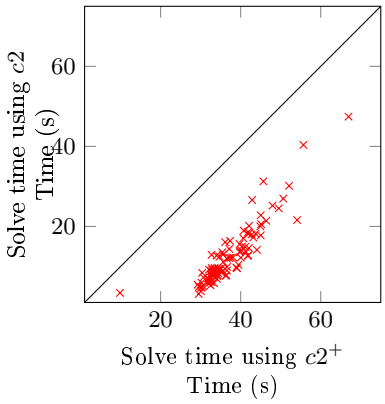

Figure 11. Anomaly 4: scatter-plot of Z3 with tighter lower bound ($c2^+$) and without lower bound ($c2$) in the basic model for the unsatisfiable case.

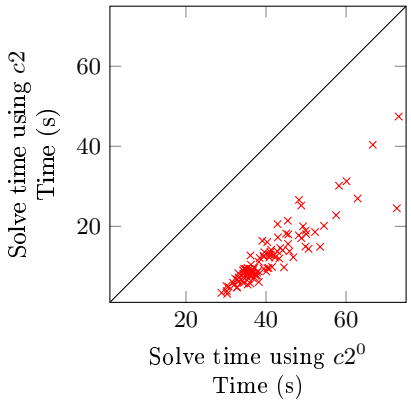

Figure 12. Anomaly 4: scatter-plot of Z3 with lower bound greater than 0 ($c2^0$) and without lower bound ($c2$) in the basic model for the satisfiable case.

smaller instances. Nonetheless, for one solver the solution time is not monotonic in the size of the problem instance.

In Anomaly 3 a trivial change to a model, that would have little to no effect on a CP solver, can greatly reduce the solve time of one instance while greatly increasing that of another instance. Though we would expect any change in performance to be a very small negative one, we see an average *improvement* of 20% with two of the solvers.

Finally, Anomaly 4 is perhaps the most baffling to anyone trying to build better models. Here we see a case where tightening bounds on a variable impedes performance even in unsatisfiable instances.

In seeking an explanation of these anomalies one must start with the observation that the behaviour of SMT solvers can be chaotic; a small change in the model can result in a large, seemingly random change in the behaviour of the solver. For example, it is well known that changing the order of the constraints in an SMT model can greatly increase or decrease the solution time.

This chaos is certainly present in Anomaly 3 where we see the addition of an extra variable can greatly increase or decrease solution time. However, we believe that beneath the noise there is still a significant difference in the average solution time. To assert this with greater confidence we would need to gather a large sample size and conduct a statistical test.

It is difficult to see how chaos could play a significant role in the other anomalies, especially so in Anomoly 4 where the tighter bounds increases solution. This increase is observed on 100 out of 100 satisfiable instances and 100 out of 100 unsatisfiable instances. And this behaviour is repeated for two forms of tighter bounds. So the phenomenon is robust, being observed 100 times out of 100 in each of 4 different settings. The anomaly arises in spite of chaos, not because of chaos.

Perhaps there are explanations for all these behaviours, but, from what we currently know, developing an understanding of what makes a good SMT(QF_LIA) model appears to be an enormous challenge. Anyone working on SMT modelling of combinatorial problems needs to proceed cautiously, expect the unexpected and be prepared for difficulties.

In the future we hope to work with the developers of SMT solvers to try to develop an understanding of how modelling choices affect solver performance and to consider whether SMT solvers can be tuned to perform better on combinatorial problems. In future experiments we plan to reduce the affect of chaos by averaging over large samples of instances that have their constraints randomly ordered.

Acknowledgement

We thank James Cussens and Mark Bartlett for explaining the pedigree reconstruction problem and supplying us with their problem instance generator. We are grateful to the referees of this paper for their valuable feedback. Miquel Palahí has been supported by the Spanish Ministry of Science and Innovation (project

TIN2012-33042) the Universitat de Girona (grant BR 2010) and the Economy and Knowledge Department (Generalitat de Catalunya) and SUR (BE-DGR 2012).

References

1. M. Bofill, M. Palahí, J. Suy, and M. Villaret. Solving constraint satisfaction problems with SAT modulo theories. *Constraints*, 17(3):273–303, 2012.
2. A. Cimatti, A. Griggio, B. J. Schaafsma, and R. Sebastiani. The MathSAT5 SMT solver. In N. Piterman and S. A. Smolka, editors, *TACAS*, volume 7795 of *Lecture Notes in Computer Science*, pages 93–107. Springer, 2013.
3. R. G. Cowell. Efficient maximum likelihood pedigree reconstruction. *Theoretical Population Biology*, 76(4):285–291, 2009.
4. J. Cussens, M. Bartlett, E. M. Jones, and N. A. Sheehan. Maximum Likelihood Pedigree Reconstruction Using Integer Linear Programming. *Genetic Epidemiology*, 37(1):69–83, January 2013.
5. L. M. de Moura and N. Bjørner. Z3: An Efficient SMT Solver. In *Proceedings of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2008)*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.
6. B. Dutertre and L. de Moura. The Yices SMT solver. Tool paper available at http://yices.csl.sri.com/tool-paper.pdf, August 2006.