ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230520

Unit Refutations of Difference Constraint Systems

K. Subramani[a;*] **and Piotr Wojciechowski**[a]

[a]LDCSEE, West Virginia University, Morgantown, West Virginia, USA

Abstract. This paper is concerned with a refutation system (proof system) for a class of linear constraint systems called difference constraint systems (DCS). In particular, we study the refutability of DCSs in the unit refutation (UR) system. Recall that a difference constraint is a linear relationship of the form: $x_i - x_j \leq b_{ij}$ and a DCS is a conjunction of such constraints. Associated with a refutation system are three important features, viz., (a) Soundness, (b) Completeness, and (c) Efficiency. The UR system is sound and efficient; however, it is **incomplete**, in that unsatisfiable DCSs may not have unit refutations. We establish that this refutation system is efficient in that there exists a tractable algorithm for determining if a DCS has a UR. Investigating **weak** (incomplete) refutation systems leads to a better understanding of the inference rules required for establishing contradictions in the given constraint system. Thus, this study is well-motivated. Despite the fact that unit refutations can be exponentially long in terms of the input system size, we provide a compact representation of these refutations. This compact representation is an important contribution of this paper.

1 Introduction

This paper is concerned with a restricted refutation system for Difference Constraint Systems (DCSs). Recall that a difference constraint is a linear relationship of the form: $x_i - x_j \leq b_{ij}$. A conjunction of such constraints, and constraints of the form $x_i \leq b_i$ or $-x_i \leq b_i$, is called a DCS and can be written in matrix form as: $\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$. DCSs occur in a number of application domains such as abstract interpretation [5, 6] and program verification [5, 4].

Refutations can be thought of as negative certificates. Certificates (and certifying algorithms) enhance the reliability of software. This is true even for **incomplete** refutation systems, such as read-once refutations [9]. It is worth noting that enforcing the read-once requirement makes refutations easier to visualize. In a read-once refutation, each constraint is used at most once in an inference step. This makes visualization realizable.

In a unit refutation, an absolute constraint (one variable constraint) **must** be used in each inferential step. Unit Refutations (URs) in DCSs are interesting because they are domain specific refutations. In other words, they prove that a DCS is infeasible using its current set of absolute (one variable) constraints without necessarily proving the infeasibility of the underlying system of relative (two variable) constraints. This is a marked difference from unrestricted refutations. Our goal in this paper is to investigate the algorithmic complexity of finding negative certificates for DCSs within the UR refutation system.

* Corresponding Author. Email: k.subramani@mail.wvu.edu.

Observe that UR is an incomplete refutation system. However, incomplete systems have been studied extensively in propositional proof complexity. For instance, [9] details the computational complexity of read-once refutations in CNF formulas. Likewise, [10] discusses read-once refutations in Horn formulas. It is curious that unit resolution is complete for Horn formulas, but incomplete for arbitrary CNF formulas [7]. Read-once refutation systems have also been studied for linear constraint systems [20]. Note that a DCS may have a unit refutation that is exponentially long in terms of the size of the input system (see Theorem 1). However, in this paper, we provide a compact certificate for establishing that a DCS has a unit refutation (see Theorem 2).

The principal contributions of this paper are as follows:

1. The design of a polynomial time algorithm for checking if a DCS has a unit refutation (Section 5).
2. Establishing that the problem of determining a shortest tree-like unit refutation of a DCS is **NP-hard** (Section 6).
3. A $O(m \cdot n^2 \cdot ||\mathbf{b}||_\infty)$ time algorithm for finding a shortest tree-like unit refutation of a DCS with n variables (Section 6).
4. A 2-approximation algorithm for the problem of finding a shortest tree-like unit refutation of a DCS (Section 7).

2 Statement of Problems

In this section, we formally describe the problems under consideration.

Definition 2.1 *A difference constraint is a constraint of the form $a_i \cdot x_i + a_j \cdot x_j \leq b_k$ where $a_i \neq a_j \in \{1, 0, -1\}$ and $b_k \in \mathbb{Z}$.*

In a difference constraint, we refer to b_k as the defining constant. Additionally, the terms x_i and $-x_i$ are referred to as literals. If a difference constraint has only one non-zero coefficient, then it is called an absolute constraint.

Example (1): The following are difference constraints:

$$x_1 - x_2 \leq 4 \qquad x_2 - x_3 \leq 5 \qquad x_4 \leq -1.$$

Furthermore, $x_4 \leq -1$ is an absolute constraint.

Definition 2.2 *A conjunction of difference constraints is known as a Difference Constraint System (DCS).*

A DCS can be represented as a matrix $\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$. In this representation, $\mathbf{b}$ is referred to as the defining constant vector. Unless otherwise stated, we use n to refer to the number of variables in a DCS and m to refer to the number of constraints.

Example (2): System (1) is a DCS with 2 constraints over 3 variables.

$$x_1 - x_2 \leq 5 \qquad - x_2 + x_3 \leq -2. \tag{1}$$

In this paper, we are focused on proving that a DCS does not have any linear (rational) solutions. This is the linear refutability problem (LRP). Such a proof of linear infeasibility is known as a negative certificate or refutation. For LRP, these refutations consist of a sequence of inferences that results in a contradiction of the form $0 \leq b$ where $b < 0$. In this case, we use a single inference rule known as the ADD rule (given by Rule 2).

$$\text{ADD} : \frac{\sum_{i=1}^{n} a_i \cdot x_i \leq b_1 \qquad \sum_{i=1}^{n} a'_i \cdot x_i \leq b_2}{\sum_{i=1}^{n} (a_i + a'_i) \cdot x_i \leq b_1 + b_2} \tag{2}$$

It is easy to see that Rule (2) is sound since any assignment that satisfies the hypotheses also satisfies the consequent. From Farkas' lemma [8], we know that the rule is complete as well.

Example (3): Consider the constraints $x_1 - x_2 \leq -1$ and $x_2 - x_3 \leq 4$. Applying the ADD rule to these constraints results in the constraint $x_1 - x_3 \leq 3$.

Definition 2.3 *A sequence of applications of the ADD rule that results in a contradiction is known as a linear refutation.*

In this paper, we study a restricted version of the ADD rule, known as the unit-ADD (UADD) rule. In the UADD rule, at least one of the two hypotheses in Rule (2) must be an absolute constraint. The application of the UADD rule always cancels a variable. Rule (3) represents the UADD rule when applied to difference constraints.

$$\text{UADD} : \frac{a_i \cdot x_i \leq b_1 \qquad a_j \cdot x_j - a_i \cdot x_i \leq b_2}{a_j \cdot x_j \leq b_1 + b_2} \tag{3}$$

In Rule (3), it is understood that $a_i, a_j \in \{1, -1\}$.

Example (4): Consider the constraints $x_1 \leq -1$ and $-x_1 + x_2 \leq 4$. Applying the UADD rule to these constraints results in the constraint $x_2 \leq 3$.

A linear refutation using only the UADD rule is called a unit linear refutation. The problem of finding such a refutation is called the unit linear refutability problem (ULRP). It is important to note that, unlike the ADD rule, the UADD rule is incomplete. This means that there are DCSs with no linear solutions that do not have a refutation using only the UADD rule.

Example (5): Let $\mathbf{D}$ be the following DCS:

$$x_1 - x_2 \leq 1 \qquad x_2 - x_1 \leq -2 \qquad x_1 \leq -1.$$

Since applying the ADD rule to the constraints $x_1 - x_2 \leq 1$ and $x_2 - x_1 \leq -2$ results in the constraint $0 \leq -1$, we have that $\mathbf{D}$ is infeasible. However, $\mathbf{D}$ does not have a unit refutation. Observe the following:

1. $x_1 \leq -1$ is the only absolute constraint in $\mathbf{D}$.
2. First, we must apply the UADD rule to $x_1 \leq -1$ and $x_2 - x_1 \leq -2$ to get $x_2 \leq -3$.
3. Next, we must apply the UADD rule to $x_2 \leq -3$ and $x_1 - x_2 \leq 1$ to get $x_1 \leq -2$.

Note that there is no way to cancel x_1 from a constraint of the form $x_1 \leq b$ or x_2 from a constraint of the form $x_2 \leq b$ without introducing the term x_2 or x_1, respectively. Thus $\mathbf{D}$ does not have a unit linear refutation.

In this paper, we also study tree-like unit refutations.

Definition 2.4 *A* **tree-like** *refutation is a refutation in which each derived constraint can be used at most once.*

Observe that tree-like refutations can reuse the input constraints (i.e., constraints in the original DCS). However, non-input constraints must be re-derived, if necessary.

Example (6): Let $\mathbf{D}$ be the constraint system in System (4).

$$\begin{aligned} x_1 \leq 1 \qquad & - x_1 + x_2 \leq -1 \\ x_1 - x_2 \leq 0 \qquad & - x_1 \leq -1 \end{aligned} \tag{4}$$

A tree-like unit refutation of $\mathbf{D}$ is shown in Figure 1.

Definition 2.5 *The* **length** *of a tree-like refutation is the number of inferences in the refutation.*

Using this notion of length, we can define the problem of finding a shortest tree-like refutation.

Definition 2.6 *An* **Optimal Length Tree-like Unit Refutation (OTLUR)** *of a DCS is a tree-like unit refutation with minimum length.*

Example (7): The tree-like unit refutation of System (4) in Example (6) consists of 3 inferences. Thus, it is a refutation of length 3. Observe that it is also an OTLUR of System (4).

We now show that the length of a tree-like unit refutation can be exponential in the size of the DCS. Thus, we cannot construct a tree-like refutation inference by inference.

Theorem 1 *The OTLUR of a DCS with n variables can have length at least $(n \cdot (f(n) + 1) + 2)$ for an arbitrary function $f(n)$.*

Proof: Let $\mathbf{D}$ be the DCS in System (5):

$$\begin{array}{rcl|rcl} x_1 & \leq & f(n) & x_2 - x_3 & \leq & 0 \\ -x_1 & \leq & 0 & & \cdots & \\ x_1 - x_2 & \leq & -1 & x_{n-1} - x_n & \leq & 0 \\ & & & x_n - x_1 & \leq & 0. \end{array} \tag{5}$$

Let R be a tree-like unit refutation of $\mathbf{D}$. Note that any unit refutation of $\mathbf{D}$ must use the constraint $x_1 \leq f(n)$. To derive the constraint $x_1 \leq f(n) - 1$, R needs to use the constraint $x_1 - x_2 \leq -1$. To, derive a constraint of the form $x_2 \leq b$ from $x_1 \leq f(n)$, R needs to use the constraints $x_n - x_1 \leq 0$ through $x_2 - x_3 \leq 0$. This uses $(n-1)$ constraints. Adding the constraint $x_1 - x_2 \leq -1$ to this constraint results in the constraint $x_1 \leq f(n) - 1$.

Consequently, deriving $x_1 \leq -1$ through the unit ADD rule requires a total of $(n \cdot (f(n) + 1) + 1)$ constraints. To derive the final contradiction we need to use the constraint $-x_1 \leq 0$. Thus, any unit refutation of $\mathbf{D}$ needs to use at least $(n \cdot (f(n) + 1) + 2)$ constraints.

It follows that if $f(n)$ is chosen to be an exponential function of n, then the optimal length tree-like unit refutation will have exponential length. □

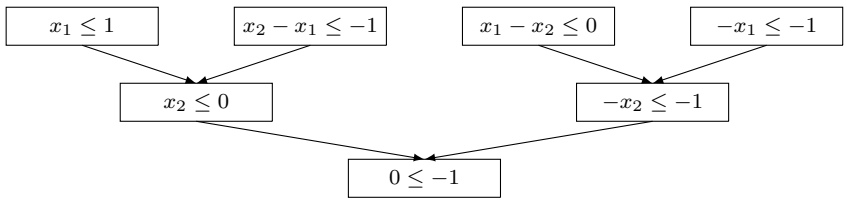

Figure 1. Tree-like Unit Refutation

In this paper, we represent DCSs as constraint networks, as described in [3] (pages 666 and 667). The construction consists of the following steps:

1. For each variable x_i, create the vertex x_i. Additionally, create the vertex x_0.
2. For each constraint of the form $x_i - x_j \leq b_{ij}$, create the edge $x_j \overset{b_{ij}}{\rightarrow} x_i$.
3. For each constraint of the form $x_i \leq b_i$, create the edge $x_0 \overset{b_i}{\rightarrow} x_i$.
4. For each constraint of the form $-x_j \leq b_j$, create the edge $x_j \overset{b_j}{\rightarrow} x_0$.

Observe that the above construction establishes $\mathbf{AC^0}$-equivalence between the linear feasibility problem in DCSs and the shortest path problem in real-weighted networks. This equivalence is exploited in Theorem 2, Theorem 3, and Theorem 6.

3 Motivation

As mentioned before, refutations certify negative answers. Certificates enhance confidence in the results of computations [13]. In general, generating refutations is a non-trivial task. This is particularly true in case of **NP-hard** problems. Indeed, in case of **NP-hard** problems, refutations tend to be exponential in the size of the input. One way around this hurdle is to focus on **incomplete** refutation systems. An incomplete refutation system typically provides refutations that are polynomial in the size of the input; however, it may be the case that some unsatisfiable instances lack refutations.

In this paper, we focus on unit refutations only. In a unit refutation, each inference must use an absolute constraint. The focus on unit refutations in this paper stems from a fundamental difference between absolute constraints and relative constraints in DCSs. Since absolute constraints only place bounds on a single variable, they can be used to define the domain over which feasibility is considered. Meanwhile, relative constraints define the relationship between variables and can be considered domain agnostic. This difference in the two types of difference constraints carries over to create a difference between unit and non-unit refutations.

A unit refutation relies on the absolute constraints in the underlying constraint system. Thus, a unit refutation serves as a domain specific refutation. This is in contrast to an unrestricted refutation which may be domain agnostic. Thus, a study of unit refutations is important since it reveals the structure of such domain specific refutations. Additionally, unit refutations are refutations that would be encountered while using unit propagation. Unit propagation is used in many abstract solvers [15].

A restriction that makes a proof system incomplete can guarantee short refutations when otherwise refutations may be exponential in the size of the input. For example, refutations of **NP-hard** problems can be exponentially long. In fact, if there exists a complete refutation system which always generates polynomially sized refutations for an **NP-hard** problem, then **NP = coNP**. Thus, any refutation system which is guaranteed to generate short refutations for **NP-hard** problems is likely to be incomplete. It follows that incomplete refutation systems are important when short refutations are required. While a refutation system may be incomplete, the refutation system might still be complete for the set of instances that correspond to real-world problems.

4 Related Work

In typical constraint systems, certificates can be divided into two main categories, positive certificates and negative certificates. For a given constraint system $\mathbf{D}: \mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$, any satisfying assignment to the system serves as a positive certificate. However, the form negative certificates take depends on the form of the refutation system.

The infeasibility of linear systems is commonly established using Farkas' lemma [8, 12]. As a result of Farkas' lemma, a proof of linear infeasibility of $\mathbf{D}$ can simply be a non-negative vector $\mathbf{y}$, such that $\mathbf{y} \cdot \mathbf{A} = \mathbf{0}$, $\mathbf{y} \cdot \mathbf{b} < \mathbf{0}$. This vector $\mathbf{y}$ is called the Farkas witness of the infeasibility of $\mathbf{D}$. Similarly, the elements of $\mathbf{y}$ are called Farkas variables.

Observe that the Farkas vector $\mathbf{y}$ corresponds to a weighted summation of the constraints in $\mathbf{D}$. Such a refutation can be broken up into individual summations, each between a pair of constraints. These summations correspond to applications of the ADD inference rule. This provides an additional form for refutations of linear programs which corresponds to a step by step refutation procedure.

This paper focuses specifically on refutations for DCSs. In [18], the problem of finding the shortest linear refutation was studied. It was shown that this problem was equivalent to the problem of finding a negative cycle with the fewest number of edges in a directed, weighted network. They also designed an algorithm that runs in $O(n^3 \cdot \log K)$ time, where n is the number of vertices in the corresponding network $\mathbf{G}$ [3], and K is the length of the refutation. The current fastest algorithm runs in $O(m \cdot n \cdot K)$ time [19], where m is the number of edges in $\mathbf{G}$. Recently, a randomized algorithm for this problem was proposed [1].

Unit refutations have also been studied for Horn constraint systems (HCSs) [23]. In [23], it was shown that the problem of determining if an HCS has a unit refutation is **NP-complete**. Additionally, it was shown that the OTLUR problem for HCSs is **NPO-complete**. Finally, it was shown that the problem of determining if an HCS has a unit refutation is unlikely to have a kernel whose size is polynomial in the length of the refutation.

An even more restricted form of unit refutation for DCSs was studied in [22]. In this form of unit refutation, known as read-once unit refutation, each constraint could be used at most once. In [22], it was

shown that the problem of checking if a DCS has a read-once unit refutation is **NP-complete**. It was also shown that the problem of finding a shortest read-once unit refutation is **NPO-complete**. [22] also described both fixed-parameter tractable and exact exponential algorithms for finding read-once unit refutations of DCSs.

5 Unit Refutations in Difference Constraint Systems

In this section, we examine the problem of finding unit refutations of DCSs. We utilize the graph construction in [3]. As mentioned in [3], each DCS corresponds to a weighted directed graph. In this graph, each vertex corresponds to a variable and each edge corresponds to a constraint. Additionally, there is a vertex x_0 used to handle absolute constraints. We now show that a DCS $\mathbf{D}$ has a unit refutation, if and only if, the corresponding graph $\mathbf{G}$ has a negative weight closed walk that uses the vertex x_0. Unlike a simple cycle, this walk can reuse edges and vertices.

Lemma 5.1 *A DCS $\mathbf{D}$ has a unit refutation, if and only if, the corresponding graph $\mathbf{G}$ has a negative weight closed walk using x_0.*

Proof: First, suppose that $\mathbf{D}$ has a unit refutation R. Recall that each application of the UADD rule has the following form:

$$\frac{a_i \cdot x_i \leq b_i \qquad\qquad -a_i \cdot x_i + a_j \cdot x_j \leq b_{ij}}{a_j \cdot x_j \leq b_i + b_{ij}}$$

No two-variable constraints can be derived in this way. If a non-absolute constraint is derived, then it has to be of the form $0 \leq b$. If $b < 0$, then we have derived a contradiction and thus produced a refutation. If $b \geq 0$, then the remaining portion of R must be a unit refutation. Thus, without loss of generality, we can assume that only the final application of the UADD rule in R derives a non-absolute constraint.

We can assume without loss of generality that the first inference in R is

$$\frac{a_i \cdot x_i \leq b_i \qquad\qquad a_k \cdot x_k - a_i \cdot x_i \leq b_{ki}}{a_k \cdot x_k \leq b_i + b_{ki}}$$

and that the last inference is

$$\frac{a_j \cdot x_j \leq b - b_j \qquad\qquad -a_j \cdot x_j \leq b_j}{0 \leq b.}$$

From R, we can construct a negative weight closed walk C in $\mathbf{G}$ as follows:

1. Start with the constraint $a_i \cdot x_i \leq b_i$. If $a_i = 1$, then by construction of $\mathbf{G}$, the edge $x_0 \stackrel{b_i}{\rightarrow} x_i$ is in $\mathbf{G}$. Add this edge to C.
   If $a_i = -1$, then by construction of $\mathbf{G}$, the edge $x_i \stackrel{b_i}{\rightarrow} x_0$ is in $\mathbf{G}$. Add this edge to C.
2. Let $a_k \cdot x_k - a_i \cdot x_i \leq b_{ki}$ be the other constraint used by the first application of the UADD rule. Add the corresponding edge of $\mathbf{G}$ to C.
3. Let $a_{i'} \cdot x_{i'} + a_{j'} \cdot x_{j'} \leq b_{i'j'}$ be the a constraint used by the r^{th} application of the UADD rule. Add the corresponding edge of $\mathbf{G}$ to C.
4. Finally, consider the constraint $-a_j \cdot x_j \leq b_j$. If $a_j = 1$, then by construction of $\mathbf{G}$, the edge $x_j \stackrel{b_j}{\rightarrow} x_0$ is in $\mathbf{G}$. Add this edge to C.
   If $a_i = -1$, then by construction of $\mathbf{G}$, the edge $x_0 \stackrel{b_j}{\rightarrow} x_j$ is in $\mathbf{G}$. Add this edge to C.

From the structure of R, the constraint introduced by the r^{th} inference must cancel the variable introduced by the $(r-1)^{th}$ inference. Thus, these constraints share a variable $x_{i'}$ and the coefficients of $x_{i'}$ in these constraints have opposite signs. These constraints correspond to the $(r+1)^{th}$ and r^{th} edges in C respectively. Thus, C is a valid path from x_0 to itself.

By construction, the total weight of all the edges in C is $b < 0$. Thus, C is a negative weight closed walk through x_0.

Now suppose that $\mathbf{G}$ has a negative weight closed walk C through the vertex x_0. We can reverse the process used to construct C to construct a unit refutation R. □

Now we need a way to detect if $\mathbf{G}$ contains a negative weight closed walk through the vertex x_0. Recall that the length of this walk can be exponential in the size of the input.

This can be done in polynomial time by looking at the strongly connected components of $\mathbf{G}$.

Lemma 5.2 *If $\mathbf{D}$ has a unit refutation, then the strongly connected component of $\mathbf{G}$ containing x_0 contains a negative cycle.*

Proof: Assume that $\mathbf{D}$ has a unit refutation. From Lemma 5.1, $\mathbf{G}$ contains a negative weight closed walk C through the vertex x_0. Let x_i be a vertex on this cycle.

By the construction of C in Lemma 5.1, the constraints $x_i \leq b_i$ and $-x_i \leq b_i'$ must be derivable from $\mathbf{D}$ by unit refutation. If we repeat the process used to construct C using the derivation of $x_i \leq b_i$, we obtain a walk from x_0 to x_i. If we repeat the process used to construct C using the derivation of $-x_i \leq b_i'$, we obtain a walk from x_i to x_0. Thus, x_i is in the strongly connected component of $\mathbf{G}$ containing x_0.

Since x_i was an arbitrary vertex on C, every vertex on C must be in the strongly connected component of $\mathbf{G}$ containing x_0. Thus, this strongly connected component contains a negative cycle. □

Lemma 5.3 *If the strongly connected component of $\mathbf{G}$ containing x_0 contains a negative cycle, then $\mathbf{D}$ has a unit refutation.*

Proof: Assume that the strongly connected component of $\mathbf{G}$ containing x_0 contains a cycle C' of weight $b_{C'} < 0$. Let x_i be a vertex on this cycle. Since x_i is in the strongly connected component of $\mathbf{G}$ containing x_0, there must be a path p_1 from x_0 to x_i. Let b_1 be the weight of this path. Similarly, there must be a path p_2 from x_i to x_0. Let b_2 be the weight of this path.

Thus we can construct a negative cycle C as follows: 1. Add the path p_1 to C. 2. Add $\left\lfloor \frac{b_1+b_2}{-b_{C'}} + 1 \right\rfloor$ copies of the negative cycle C' to C. 3. Add the path p_2 to C. By construction, C is a closed walk through the vertex x_0. Now we need to show that C has negative weight.

The weight of C is $b_1 + b_2 + \left\lfloor \frac{b_1+b_2}{-b_{C'}} + 1 \right\rfloor \cdot b_{C'} \leq b_1 + b_2 + \left(\frac{b_1+b_2}{-b_{C'}} + 1 \right) \cdot b_{C'} = b_{C'} < 0$. Thus C is a negative weight closed walk through x_0 as desired. □

From Lemmas 5.2 and 5.3, we have the following result.

Theorem 2 $\mathbf{D}$ *has a unit refutation, if and only if, the strongly connected component $\mathbf{G}'$ of $\mathbf{G}$ containing x_0 contains a negative cycle.*

Proof: If $\mathbf{D}$ has a unit refutation, then by Lemma 5.2, $\mathbf{G}'$ contains a negative cycle. If $\mathbf{G}'$ contains a negative cycle, then by Lemma 5.3, $\mathbf{D}$ has a unit refutation. □

From Lemmas 5.2 and 5.3, we have the following result.

Theorem 3 *The problem of checking if a DCS* $\mathbf{D}$ *with* m *constraints over* n *variables has a unit refutation can be solved in* $O(m \cdot n)$ *time.*

Proof: From Lemmas 5.2 and 5.3 we know that $\mathbf{D}$ has a unit refutation, if and only if, the strongly connected component $\mathbf{G}'$ of $\mathbf{G}$ containing x_0 contains a negative cycle.

Observe that $\mathbf{G}'$ can be found in $O(m+n)$ time. Once $\mathbf{G}'$ is found checking $\mathbf{G}'$ contains a negative cycle can be done in $O(m \cdot n)$ time. □

The connected component and negative cycle described in the proof of Theorem 3 serve as a certificate that a DCS has a unit refutation. Note that this is not the unit refutation itself. However, it is a polynomial sized, polynomial checkable way to verify that such a certificate exists.

6 Shortest Tree-like Unit Refutations

In this section, we show that the problem of finding the length of an OTLUR of a DCS is **NP-hard**.

We do this by a reduction from the 1-neighbor knapsack problem (1KK) problem. This problem is known to be **NP-hard** even with unit weights and when the input graph is a DAG [2].

The 1KK problem is defined as follows: Let $\mathbf{G} = \langle \mathbf{V}, \mathbf{E}, w, p \rangle$ be a directed graph with vertex set $\mathbf{V}$, edge set $\mathbf{E}$, vertex weight function w, and vertex profit function p. For each vertex $v \in V$ let $I(v) = \{v' : (v', v) \in \mathbf{E}\}$ be the set of vertices that are the endpoints of edges incoming to v.

Given target weight W, and target profit P, does there exist a set of vertices S such that $\sum_{v \in S} w(v) \leq W$, $\sum_{v \in S} p(v) \geq P$, and if $v \in S$ and $I(v) \neq \emptyset$, then $I(v) \cap S \neq \emptyset$. A set S that satisfies all of these conditions is called feasible.

This problem is **NP-hard** even when $w(v) = 1$ for all vertices in $\mathbf{G}$ and $\mathbf{G}$ is a directed acyclic graph [2].

Let $\mathbf{G}$ be a DAG. Additionally, let p be a profit function on the vertices of $\mathbf{G}$, let P be the target profit, and let W be the target weight.

From $\mathbf{G}$, we construct a DCS $\mathbf{D}$ as follows:

1. For each vertex v_i in $\mathbf{G}$, create the variable x_i.
2. Create the variable x_0 and the constraints $x_0 \leq P - 1$ and $-x_0 \leq 0$.
3. For each vertex v_i such that $I(v_i) = \emptyset$, create the constraints $x_0 - x_i \leq -p(v_i)$ and $x_i - x_0 \leq 0$.
4. For each edge (v_i, v_j), create the constraints $x_i - x_j \leq -p(v_j)$ and $x_j - x_i \leq 0$.

We will now show that there exists a feasible set S of vertices, if and only if, $\mathbf{D}$ has a unit refutation of length at most $(2 \cdot W + 2)$.

Theorem 4 *The OTLUR problem for DCSs is* **NP-hard**.

Proof: Let $\mathbf{G}$ be a DAG and let $\mathbf{D}$ be the corresponding DCS.

First, assume that $\mathbf{G}$ has a feasible set S of vertices. From S, we construct a set R of constraints as follows:

1. Add the constraints $x_0 \leq P - 1$ and $-x_0 \leq 0$ to R.
2. For each vertex $v_i \in S$:
   (a) If $I(v_i) = \emptyset$, then add the constraints $x_0 - x_i \leq -p(v_i)$ and $x_i - x_0 \leq 0$ to R.
   (b) If $I(v_i) \neq \emptyset$, then let v_j be a vertex in $I(v_i) \cap S$. Since S is a feasible set of vertices and $I(v_i) \neq \emptyset$, then $I(v_i) \cap S \neq \emptyset$. Thus, such a vertex is guaranteed to exist. Add the constraints $x_j - x_i \leq -p(v_i)$ and $x_i - x_j \leq 0$.

Since $\mathbf{G}$ is a DAG, $\mathbf{G}$ must have vertices with no incoming edges. Consider a vertex $v_i \in S$. If v_i has incoming edges, then there exists a vertex $v_j \in S$ such that the edge $(v_j, v_i) \in \mathbf{G}$. The same argument applies to v_j. Since $\mathbf{G}$ is a DAG, at least one vertex with no predecessors must be in S. Thus R contains an absolute constraint. By construction of R, each literal x_i appears the same number of times as the literal $-x_i$. Thus, summing the constraints in R results in a constraint of the form $0 \leq b$.

For each vertex in S, we added one constraint to R with defining constant $-p(v_i)$. Additionally, we added a constraint to R with defining constant $(P-1)$. By construction, all other constraints in R have defining constant 0. Thus, summing the constraints in R results in a constraint with defining constant $b = P - \sum_{v_i \in S} p(v_i)$. Since S is feasible, $\sum_{v_i \in S} p(v_i) \geq P$. Thus, $b < 0$. Consequently, R is a unit refutation of $\mathbf{D}$. Observe that R has $(2 \cdot |S| + 2)$ constraints. Since $w(v_i) = 1$ for each vertex v_i, $(2 \cdot |S| + 2) = 2 \cdot \sum_{v_i \in S} w(v_i) + 2$. Since S is feasible $\sum_{v_i \in S} w(v_i) \leq W$. Thus R has length at most $(2 \cdot W + 2)$ as desired.

Now assume that $\mathbf{D}$ has a tree-like unit refutation R of length $(2 \cdot W + 2)$. From R we construct a set S of vertices as follows: For each constraint of the form $x_j - x_i \leq -p(v_i)$ in R add the vertex v_i to S.

Consider the vertex v_i added due to the constraint $x_j - x_i \leq -p(v_i)$. Since R is a unit refutation, a constraint of the form $x_j \leq b$ must be derivable from the constraints used by R. Thus, R must contain a constraint of the form $x_k - x_j \leq -p(v_j)$. Thus, v_j is in S. Thus, a vertex v_i is in S only if a predecessor of S is in S.

For each vertex in S, there are at least two constraints in R. Additionally, R contains at least two absolute constraints. Thus, S has at most W vertices.

The only absolute constraints in $\mathbf{D}$ are $x_0 \leq P - 1$ and $-x_0 \leq 0$. Thus these constraints are in R. The only other constraints in R with non-zero defining constants are of the form $x_j - x_i \leq -p(v_i)$. Each vertex v_i of profit $p(v_i)$ in S corresponds to such an edge. Thus, $\sum_{v \in S} p(v) \geq P$. Consequently, S is a set of vertices that satisfies the conditions required by 1KK. □

We utilize the structure described in Lemma 5.3 to prove an upper bound on the lengths of tree-like unit refutations of DCSs. In the theorems in this section, b_{max} represents the largest defining constant in a DCS.

Theorem 5 *Let* $\mathbf{D}$ *be a DCS with* n *variables that has a UR. The OTLUR of* $\mathbf{D}$ *has length at most* $(2 \cdot n^2 \cdot b_{max} + 3 \cdot n)$.

Proof: Let $\mathbf{D}$ be a DCS with a unit refutation. Additionally, let $\mathbf{G}$ be the graph corresponding to $\mathbf{D}$. From Lemma 5.2, the strongly connected component of $\mathbf{G}$ containing x_0 contains a negative cycle.

From Lemma 5.3, $\mathbf{D}$ has a unit refutation that corresponds to a simple path p_1, a possibly repeated negative cycle C' and a simple path p_2. Each of these uses no vertex more than once. Thus, p_1, p_2 and C' each have at most n edges.

Since b_{max} is the largest edge weight in $\mathbf{G}$. The weight b_1 of p_1 is at most $n \cdot b_{max}$. Similarly, the weight b_2 of p_2 is at most $n \cdot b_{max}$. Since, C' is a negative cycle the weight $b_{C'}$ of C' is at most -1.

To, make a negative weight closed walk C through x_0, we need $\left\lfloor \frac{b_1+b_2}{-b_{C'}} + 1 \right\rfloor$ copies of C'. Note that $\left\lfloor \frac{b_1+b_2}{-b_{C'}} + 1 \right\rfloor \leq 2 \cdot n \cdot b_{max} + 1$. Thus, the number of edges in C is at most $n + n + n \cdot (2 \cdot n \cdot b_{max} + 1) = 2 \cdot n^2 \cdot b_{max} + 3 \cdot n$.

This means that $\mathbf{D}$ has a tree-like unit refutation of length at most $(2 \cdot n^2 \cdot b_{max} + 3 \cdot n)$. Thus, the OTLUR of $\mathbf{D}$ has length at most $(2 \cdot n^2 \cdot b_{max} + 3 \cdot n)$. □

We now present a $O(m \cdot n^2 \cdot ||\mathbf{b}||_\infty)$ time pseudo-polynomial time algorithm for the OTLUR problem for DCSs. $||\mathbf{b}||_\infty$ is called the infinity norm, and is the largest absolute value of any element in $\mathbf{b}$.

From Theorem 5, an OTLUR of a DCS has length at most $(2 \cdot n^2 \cdot ||\mathbf{b}||_\infty + 3 \cdot n)$, it can be found in $O(m \cdot n^2 \cdot ||\mathbf{b}||_\infty)$ time using the Bellman-Ford algorithm.

Theorem 6 *An OTLUR of a DCS can be found in $O(m \cdot n^2 \cdot ||\mathbf{b}||_\infty)$ time.*

Proof: Let $\mathbf{D}$ be a DCS and let $\mathbf{G}$ be the corresponding graph. From Theorem 5, an OTLUR of $\mathbf{D}$ has length l^* which is at most $(2 \cdot n^2 \cdot ||\mathbf{b}||_\infty + 3 \cdot n)$. Thus, there is a negative weight closed walk in $\mathbf{G}$ through x_0 with at most $(2 \cdot n^2 \cdot ||\mathbf{b}||_\infty + 3 \cdot n)$ edges. Thus, we can find a shortest negative weight closed walk through x_0 as follows:

1. After k iterations of the Bellman-Ford algorithm from x_0, we will find the least weight walk in $\mathbf{G}$ with at most k edges from x_0 to each vertex.
2. Since $\mathbf{D}$ has a unit refutation of length l^*, $\mathbf{G}$ has a negative weight closed walk through x_0 of length l^*.
3. After $(2 \cdot n^2 \cdot ||\mathbf{b}||_\infty + 3 \cdot n) \geq l^*$ iterations of the Bellman-Ford algorithm from x_0, we will find a negative weight closed walk through x_0.

This takes $O(m \cdot n^2 \cdot ||\mathbf{b}||_\infty)$ time. □

From Theorem 6, the OTLUR problem for DCSs has a pseudo-polynomial time algorithm. Additionally, from Theorem 4, the OTLUR problem for DCSs is **NP-hard**. Thus, this problem is weakly **NP-hard**.

7 Approximability

We now show that the problem of finding the length of an OTLUR for a DCS can be approximated to within a factor of 2 in polynomial time.

From Lemma 5.1, we have that a unit refutation for a DCS $\mathbf{D}$ corresponds to a negative weight closed walk in the corresponding graph $\mathbf{G}$. We will restrict ourselves to unit refutations of DCS $\mathbf{D}$ that have the structure described in the proof of Lemma 5.3. We refer to these as **simple unit refutations**.

Example (8): Figure 2 shows the structure of a simple unit refutation of a DCS.

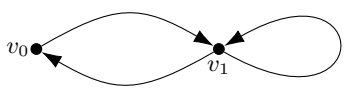

Figure 2. Structure of a simple unit refutation

Lemma 7.1 *A shortest simple unit refutation R of a DCS $\mathbf{D}$ corresponds to a negative weight closed walk W that can be divided into: 1. A simple path from v_0 to some vertex v_i. 2. A simple negative cycle through the vertex v_i, possibly repeated. 3. A simple path from v_i to v_0.*

Proof: From the proof of Lemma 5.1, we know that R corresponds to a negative weight closed walk through v_0. This walk can be divided into: 1. A walk w_1 from v_0 to some vertex v_i. 2. A closed walk w_2 through the vertex v_i, possibly repeated. 3. A walk w_3 from v_i to v_0. We now show that for some vertex v_i', these walks must be simple.

Assume for the sake of contradiction, that the walk w_1 is not simple. Thus, it contains a simple cycle C'. If this cycle has non-negative weight, then we can remove this cycle from w_1 and shorten the corresponding refutation. Thus, C' must have a weight $b_{C'} < 0$.

Consider the average weight of this cycle $\frac{b_{C'}}{|C'|}$, and compare it to the average weight of the closed walk w_2 ($\frac{b_{w_2}}{|w_2|}$). If the average weight of C' is no more than that of w_2, then we can replace loops of w_2 with loops of C' until w_2 no longer appears in the walk corresponding to R. This produces a shorter refutation. Similarly, if w_2 is more efficient than C' then we can shorten the refutation by replacing C' with another loop of w_2. In each case, the refutation can be shortened. Thus, w_1 must be a simple path. We can similarly show that w_2 and w_3 must also be simple. □

Example (9): Consider the graph in Figure 3. Consider the unit refutation corresponding to the walk $w_1 = v_0 \rightarrow v_1 \rightarrow v_4 \rightarrow v_5 \rightarrow v_1 \rightarrow v_2$. The closed walk $w_2 = v_2 \rightarrow v_3 \rightarrow v_2$, and the walk $w_3 = v_2 \rightarrow v_0$. If the walk w_2 is used once, the total weight of this walk is -1.

Note that the walk w_1 is not simple since it contains the cycle $v_1 \rightarrow v_4 \rightarrow v_5 \rightarrow v_1$. This walk can be made simple by removing this cycle from w_1. This results in path $p_1 = v_0 \rightarrow v_1 \rightarrow v_2$. We can then set $C = w_2$ and $p_2 = w_3$ to get the structure desired by Lemma 4. Note that to get the total weight of the walk consisting of p_1, C, and p_2 to be negative we need to traverse the cycle C twice. This uses a total of 7 edges which is less than the 8 edges used by the original walk.

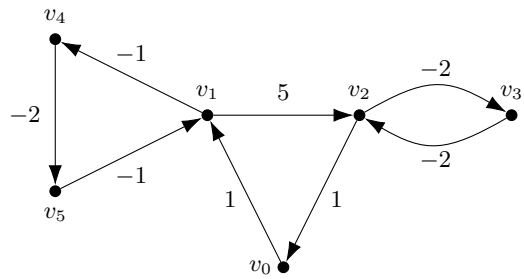

Figure 3. Example graph $\mathbf{G}$

We now show that the shortest simple unit refutation of a DCS $\mathbf{D}$ is at most twice the length of an OTLUR of $\mathbf{D}$.

Lemma 7.2 *Let $\mathbf{D}$ be a DCS with a unit refutation. Let R^* be an OTLUR of $\mathbf{D}$ and let R' be the shortest simple unit refutation of $\mathbf{D}$. The length of R' is at most twice the length of R^*.*

Proof: From the proof of Lemma 5.1, we know that R^* corresponds to a negative weight closed walk w through v_0. If w contains only one negative cycle, c, then w consists of the following: 1. A walk w_1 from v_0 to some vertex v_i on c. 2. The cycle c through v_i, possibly repeated. 3. A walk w_3 from v_i to v_0.

Since w_1 and w_3 contain no negative cycles, any cycle in w_1 or w_3 must have non-negative weight. Removing those cycles from w_3 will shorten the walk w without increasing the weight of the walk. This would contradict the assumption that w corresponds to the OTLUR R^* of $\mathbf{D}$. Thus, w_1 and w_3 must be simple. Consequently, R^* is a simple unit refutation of $\mathbf{D}$. In this case, the length of R', the shortest simple unit refutation, is at most the length of R^*. Thus, the length of R' is at most twice the length of R^*.

Now assume that R^* contains multiple negative cycles, c_1 through c_p. For each cycle c_i consider the average weight of the cycle $\frac{-b_{c_i}}{|c_i|}$. Without loss of generality assume that $\frac{-b_{c_1}}{|c_1|} \leq \frac{-b_{c_i}}{|c_i|}$ for each $i = 2, \ldots, p$. For each cycle c_i, let r_i be the number of times c_i appears in w. If, $r_1 \cdot |c_1| < \frac{\sum_{i=1}^{p} r_i \cdot |c_i|}{2}$, then we can find a shorter negative walk w' by replacing several non-c_1 negative cycles with an additional copy of c_1. This contradicts the assumption that w corresponds to the OTLUR of $\mathbf{D}$. Thus, $r_1 \cdot |c_1| \geq \frac{\sum_{i=1}^{p} r_i \cdot |c_i|}{2}$. This means that $r_1 \cdot |c_1| \geq \sum_{i=2}^{p} r_i \cdot |c_i|$. Since $\frac{-b_{c_1}}{|c_1|} \leq \frac{-b_{c_i}}{|c_i|} < 0$ for each $i = 2, \ldots, p$, $-r_1 \cdot b(c_1) \leq \sum_{i=2}^{p} -r_i \cdot b(c_i)$

Let w' be the closed walk formed by removing all negative cycles, apart from c_1, from w and adding r_1 additional copies of c_1 to w. Observe that $-2 \cdot r_i \cdot b(c_i) \leq \sum_{i=1}^{p} -r_i \cdot b(c_i)$. Thus, w' is a negative closed walk. Additionally, w' contains only one negative cycle. Thus, w' corresponds to a simple unit refutation R of $\mathbf{D}$. By construction, the length of R is at most twice the length of R^*. Since R is a simple unit refutation, the length of R' is at most the length of R. Thus, the length of R' is at most twice the length of R^* as desired. □

From Lemma 7.1 and Lemma 7.2, we get the following result.

Theorem 7 *The problem of approximating the length of an OTLUR R of a DCS $\mathbf{D}$ with m constraints over n variables to within a factor of 2 can be solved in $O(n^4)$ time.*

Proof: From Lemma 7.1, a shortest simple unit refutation R corresponds to a closed walk W that can be divided into: 1. A simple path p_1 from v_0 to some vertex v_i. 2. A simple negative cycle C through the vertex v_i, possibly repeated. 3. A simple path p_2 from v_i to v_0.

If there exists a path p_1' from v_0 to v_i that is shorter than p_1 and has fewer edges than p_1, then replacing p_1 with p_1' will result in a shorter refutation. Thus, p_1 must be a shortest path from v_0 to v_i with at most $|p_1|$ edges. A similar argument can be made for C and p_2.

Thus, to find R, for each vertex v_i and $k \leq n$, we find a shortest: 1. Path from v_0 to v_i with at most k edges. 2. Cycle through v_i with at most k edges. 3. Path from v_i to v_0 with at most k edges. Observe that these values can be found in $O(m \cdot n^2)$ time.

For each combination of v_i, p_1, C, and p_2, the length of the corresponding refutation R' can be computed as follows:

1. The path p_1 has weight b_1 and uses $|p_1|$ edges.
2. The path p_2 has weight b_2 and uses $|p_2|$ edges.
3. The cycle C, has weight $-b_C < 0$ and used $|C|$ edges.
4. The overall walk has negative weight if the cycle C is repeated $\left\lceil \frac{b_1+b_2}{b_C} \right\rceil$ times (see Lemma 5.3).
5. Thus, the walk uses a total of $(|p_1| + |p_2| + |C| \cdot \left\lceil \frac{b_1+b_2}{b_C} \right\rceil)$ edges.
6. This means that R has length $(|p_1| + |p_2| + |C| \cdot \left\lceil \frac{b_1+b_2}{b_C} \right\rceil) - 1$.

Thus, once v_i, p_1, C, and p_2 are chosen the length of the refutation corresponding to these paths can be found in constant time. For each v_i, there are at most n choices for each of p_1, p_2 and C. Thus, at most n^4 possible walks need to be tested. This can be done in $O(n^4)$ time. Let R be the unit refutation of $\mathbf{D}$ returned by this process.

Let $OPT(\mathbf{D})$ be the length of an OTLUR of $\mathbf{D}$ and let $|R|$ be the length of R. From Lemma 7.2, $|R| \leq 2 \cdot OPT(\mathbf{D})$. Thus, this is a 2-approximation. □

8 Conclusion

In this paper, we studied UR, an incomplete refutation system. We looked at two problems related to unit refutations for difference constraint systems. We designed a polynomial time algorithm for the feasibility problem, i.e., the problem of checking if a DCS has a unit refutation. We showed that the problem of finding a shortest tree-like unit refutation is **NP-hard** and designed a 2-approximation algorithm for the same. We also designed a pseudo-polynomial time algorithm for this problem.

Note that unit refutations are interesting since they provide domain specific refutations, i.e., they prove that a DCS is infeasible with its current set of absolute constraints without proving infeasibility of the underlying system of relative constraints. This is a marked difference from unrestricted linear refutations which do not naturally find such refutations. While unit refutations can be exponentially long in terms of the input constraint system, there exists a compact graphical certificate that establishes the presence of a unit refutation in a DCS.

A linear constraint form that is closely related to difference constraints is Unit Two Variable Per Inequality (UTVPI) constraint form [14, 21]. As with a difference constraint, a UTVPI constraint can have at most two non-zero coefficients belonging to the set $\{-1, 1\}$. However, unlike difference constraints, a UTVPI constraint can have two variables with coefficient 1 or two variables with coefficient -1. UTVPI constraints find applications in program verification [16], array bounds checking [11], and abstract interpretation [6]. Since UTVPI constraints generalize difference constraints, the hardness results in this paper automatically apply to UTVPI constraints. We have also been able to generalize the algorithmic results (Theorem 3 and Theorem 6) and the upper bound of Theorem 5 in this paper to UTVPI constraints.

From our perspective, the following avenues are worth pursuing:

1. Integrating our algorithms within the framework of an SMT solver such as [17] – In this paper, we introduced a polynomial time algorithm for finding unit refutations of DCSs. It would be interesting to see these algorithms compared with existing algorithms for finding refutations of these constraint systems. Additionally, with appropriate refinements it may be possible to incorporate these algorithms into existing general purpose solvers.
2. Improving the approximation algorithm or the inapproximability bound – In this paper, we provide a 2-approximation algorithm for the OTLUR problem for DCSs. Additionally, we prove that the same problem cannot be approximated to within a ratio of $(\frac{15}{14} - \epsilon)$ for any $\epsilon > 0$ unless **P = NP**. It may be possible to improve one or both of these bounds by utilizing different techniques.

Acknowledgments

This research was supported in part by the Defense Advanced Research Projects Agency through grant HR001123S0001-FP-004.

References

[1] Matthew Anderson, Matthew Williamson, and K. Subramani, 'Empirical analysis of algorithms for the shortest negative cost cycle problem', *Discrete Applied Mathematics*, **253**, 167–184, (2019).

[2] Glencora Borradaile, Brent Heeringa, and Gordon T. Wilfong, 'The knapsack problem with neighbour constraints', *J. Discrete Algorithms*, **16**, 224–235, (2012).

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, The MIT Press, Cambridge, MA, 3rd edn., 2009.

[4] Scott Cotton, Eugene Asarin, Oded Maler, and Peter Niebert, 'Some progress in satisfiability checking for difference logic.', in *FORMATS/FTRTFT*, pp. 263–276, (2004).

[5] Scott Cotton and Oded Maler, 'Fast and flexible difference constraint propagation for dpll(t).', in *SAT*, pp. 170–183. Springer, (2006).

[6] Patrick Cousot and Radhia Cousot, 'Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints', in *POPL*, pp. 238–252, (1977).

[7] William F. Dowling and Jean H. Gallier, 'Linear-time algorithms for testing the satisfiability of propositional horn formulae', *The Journal of Logic Programming*, **1**(3), 267 – 284, (1984).

[8] Gyula Farkas, 'Über die Theorie der Einfachen Ungleichungen', *Journal für die Reine und Angewandte Mathematik*, **124**(124), 1–27, (1902).

[9] K. Iwama and E. Miyano, 'Intractability of read-once resolution', in *Proceedings of the 10th Annual Conference on Structure in Complexity Theory (SCTC '95)*, pp. 29–36, Los Alamitos, CA, USA, (June 1995). IEEE Computer Society Press.

[10] Hans Kleine Büning, Piotr J. Wojciechowski, and K. Subramani, 'Read-once resolutions in Horn formulas', in *Frontiers in Algorithmics - 13th International Workshop, FAW 2019, Sanya, China, April 29 - May 3, 2019, Proceedings*, pp. 100–110, (2019).

[11] S. K. Lahiri and M. Musuvathi, 'An Efficient Decision Procedure for UTVPI Constraints', in *Proceedings of the* 5^{th} *International Workshop on the Frontiers of Combining Systems, September 19-21, Vienna, Austria*, pp. 168–183, New York, (2005). Springer.

[12] Jirí Matouek and Bernd Gärtner, *Understanding and Using Linear Programming (Universitext)*, Springer-Verlag, Berlin, Heidelberg, 2006.

[13] R. M. McConnell, K. Mehlhorn, S. Näher, and P. Schweitzer, 'Certifying algorithms', *Computer Science Review*, **5**(2), 119–161, (2011).

[14] Antoine Miné, 'The octagon abstract domain', *Higher-Order and Symbolic Computation*, **19**(1), 31–100, (2006).

[15] M. Pelleau, *Abstract Domains in Constraint Programming*, Elsevier Science, 2015.

[16] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin T. Vechev, 'An abstract domain for certifying neural networks', *PACMPL*, **3**(POPL), 41:1–41:30, (2019).

[17] SRI International, *Yices: An SMT solver*. http://yices.csl.sri.com/.

[18] K. Subramani, 'Optimal length resolution refutations of difference constraint systems', *Journal of Automated Reasoning (JAR)*, **43**(2), 121–137, (2009).

[19] K. Subramani, Matthew Williamson, and Xiaofeng Gu, 'Improved algorithms for optimal length resolution refutation in difference constraint systems', *Formal Aspects of Computing*, **25**(2), 319–341, (2013).

[20] K. Subramani and Piotr Wojciechowki, 'A polynomial time algorithm for read-once certification of linear infeasibility in UTVPI constraints', *Algorithmica*, **81**(7), 2765–2794, (2019).

[21] K. Subramani and Piotr J. Wojciechowski, 'A combinatorial certifying algorithm for linear feasibility in UTVPI constraints', *Algorithmica*, **78**(1), 166–208, (2017).

[22] K. Subramani and Piotr J. Wojciechowski, 'Analyzing unit read-once refutations in difference constraint systems', in *Logics in Artificial Intelligence - 17th European Conference, JELIA 2021, Virtual Event, May 17-20, 2021, Proceedings*, eds., Wolfgang Faber, Gerhard Friedrich, Martin Gebser, and Michael Morak, volume 12678 of *Lecture Notes in Computer Science*, pp. 147–161. Springer, (2021).

[23] K. Subramani and Piotr J. Wojciechowski, 'Tree-like unit refutations in horn constraint systems', in *Language and Automata Theory and Applications - 15th International Conference, LATA 2021, Milan, Italy, March 1-5, 2021, Proceedings*, eds., Alberto Leporati, Carlos Martín-Vide, Dana Shapira, and Claudio Zandron, volume 12638 of *Lecture Notes in Computer Science*, pp. 226–237. Springer, (2021).