Mathematical Structures in Computer Science (2023), 1–17
doi:10.1017/S0960129523000300

CAMBRIDGE UNIVERSITY PRESS

PAPER

Constrained read-once refutations in UTVPI constraint systems: A parallel perspective

K. Subramani and Piotr Wojciechowski

LCSEE, West Virginia University, Morgantown, WV, USA
Corresponding author: K. Subramani; Email: k.subramani@mail.wvu.edu

(Received 10 November 2022; revised 2 May 2023; accepted 22 July 2023)

Abstract

In this paper, we analyze two types of refutations for Unit Two Variable Per Inequality (UTVPI) constraints. A UTVPI constraint is a linear inequality of the form: $a_i \cdot x_i + a_j \cdot x_j \leq b_k$, where $a_i, a_j \in \{0, 1, -1\}$ and $b_k \in \mathbb{Z}$. A conjunction of such constraints is called a UTVPI constraint system (UCS) and can be represented in matrix form as: $\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$. UTVPI constraints are used in many domains including operations research and program verification. We focus on two variants of read-once refutation (ROR). An ROR is a refutation in which each constraint is used at most once. A literal-once refutation (LOR), a more restrictive form of ROR, is a refutation in which each literal (x_i or $-x_i$) is used at most once. First, we examine the constraint-required read-once refutation (CROR) problem and the constraint-required literal-once refutation (CLOR) problem. In both of these problems, we are given a set of constraints that must be used in the refutation. RORs and LORs are incomplete since not every system of linear constraints is guaranteed to have such a refutation. This is still true even when we restrict ourselves to UCSs. In this paper, we provide **NC** reductions between the CROR and CLOR problems in UCSs and the minimum weight perfect matching problem. The reductions used in this paper assume a CREW PRAM model of parallel computation. As a result, the reductions establish that, from the perspective of parallel algorithms, the CROR and CLOR problems in UCSs are equivalent to matching. In particular, if an **NC** algorithm exists for either of these problems, then there is an **NC** algorithm for matching.

Keywords: UTVPI constraints; matching; parallel algorithms; read-once refutation

1. Introduction

This paper examines several problems associated with linearly infeasible systems of Unit Two Variable Per Inequality (UTVPI) constraints. A linear relationship of the form: $a_i \cdot x_i + a_j \cdot x_j \leq b_k$ is called a UTVPI constraint, if $a_i, a_j \in \{0, 1, -1\}$ and $b_k \in \mathbb{Z}$. A conjunction of such constraints is called a UCS. Observe that a UCS is a specialized linear program and thus can be represented in matrix form as: $\mathbf{U} : \mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$. This means that if $\mathbf{U}$ has no linear (rational) solutions, then there exists a non-negative vector $\mathbf{y}$, such that $\mathbf{y} \cdot \mathbf{A} = \mathbf{0}$, $\mathbf{y} \cdot \mathbf{b} < 0$. This follows directly from Farkas' Lemma (Farkas 1902). The vector $\mathbf{y}$ serves as a refutation of $\mathbf{U}$. This is because this vector proves that $\mathbf{U}$ has no linear (rational) solutions. In this paper, we study a specialized form of refutation called *read-once refutation*. A read-once refutation (ROR) is a refutation that uses each constraint at most once. Thus, a refutation $\mathbf{y}$ is read-once, if each element y_i of $\mathbf{y}$ belongs to the set $\{0, 1\}$. A literal-once refutation (LOR) is a refutation that uses each literal (x_i or $-x_i$) at most once. Note that every LOR is an ROR.

Of the various forms of linear refutation, read-once refutations can be considered to be the simplest. Observe that a read-once refutation of a system **U** corresponds to a subset of the constraints of **U** that when summed together derives a contradiction. This is in contrast to more general forms of refutation where the number of times each constraint is used is important. Additionally, since coefficients are unnecessary for read-once refutations, read-once refutations are more compact than more general forms of refutation. This makes read-once refutations a highly desirable proof of infeasibility.

Unfortunately, read-once and literal-once refutations are incomplete proof systems, since there exist infeasible linear programs that do not have such refutations. In fact, this is still the case for UCSs (see Section 2). Consequently, the problems of checking if an arbitrary UCS has a read-once or literal-once refutation are still interesting.

The primary focus of this paper is variants of the LOR and ROR problems in which we are given a set of constraints that the refutation is required to use. These variants are known as the constraint-required read-once refutation (CROR) and constraint-required literal-once refutation (CLOR) problems, respectively. We provide **NC** reductions between the CROR and CLOR problems in UCSs and the decision version of the minimum weight perfect matching (MWPM_D) problem (see Section 2.3). Note that the complexity class **NC** consists of problems solvable in polylogarithmic time by a parallel machine with polynomially many processors. In total, we will provide three **NC** reductions between the CROR and CLOR problems in UCSs and the MWPM_D problem.

Together, these reductions prove that the CROR and CLOR problems in UCSs are **NC** equivalent to the MWPM_D problem. All of these reductions are created assuming the CREW PRAM model of parallel computation. This extends the work done in Subramani and Wojciechowki (2019) which provided polynomial time reductions from the ROR and LOR problems in UCSs to the problem of finding a minimum weight perfect matching in an undirected graph.

The principal contributions of this paper are as follows:

1. A proof that the CROR problem in UCSs is **NC** equivalent to the MWPM_D problem.
2. A proof that the CLOR problem in UCSs is **NC** equivalent to the MWPM_D problem.

The rest of this paper is organized as follows: Section 2 formally describes the problems under consideration. The motivation for our work and related approaches in the literature are described in Section 3. In Section 4, we provide the **NC** reductions between CROR and MWPM_D. In Section 5, we provide the **NC** reductions between CLOR and MWPM_D. We conclude in Section 6 by summarizing our contributions and identifying avenues for future research.

2. Statement of Problems

In this section, we formally describe the problems under consideration and define the terms that will be used throughout this paper.

A Linear Program (LP) is a conjunction of linear inequalities. System (1) denotes the matrix representation of a linear program.

$$\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b} \tag{1}$$

Throughout this paper, we use n to denote the number of variables in an LP and m to denote the number of constraints. Thus, in System (1), **A** has dimensions $m \times n$ and **b** is an integral m-vector.

We now define several types of constraints referred to throughout this paper.

Definition 1. *A constraint of the form $a_i \cdot x_i \leq b_k$ is called an absolute constraint, if $a_i \in \{1, -1\}$ and $b_k \in \mathbb{Z}$.*

Example (1): The constraints $x_1 \leq 2$ and $-x_2 \leq 3$ are absolute constraints.

https://doi.org/10.1017/S0960129523000300 Published online by Cambridge University Press

Definition 2. *A constraint of the form $a_i \cdot x_i + a_j \cdot x_j \leq b_k$ is called a difference constraint, if $a_i, a_j \in \{1, -1\}$, $a_i = -a_j$, and $b_k \in \mathbb{Z}$.*

Example (2): The constraints $x_1 - x_3 \leq 2$ and $-x_2 + x_4 \leq -3$ are difference constraints.

A conjunction of difference constraints is called a difference constraint system (DCS).

Definition 3. *A constraint of the form $a_i \cdot x_i + a_j \cdot x_j \leq b_k$ is called a Unit Two Variable Per Inequality (UTVPI) constraint, if $a_i, a_j \in \{0, 1, -1\}$, a_i and a_j are not both 0, and $b_k \in \mathbb{Z}$.*

Example (3): The constraints $x_1 + x_2 \leq 3$ and $-x_2 - x_3 \leq -4$ are UTVPI constraints.

A conjunction of UTVPI constraints is called a UTVPI constraint system (UCS).

In the above definitions, b_k is called the defining constant of the constraint. Note that in this paper, we require b_k to be integral. Additionally, the terms x_i and $-x_i$ are called literals.

Note that both absolute constraints and difference constraints are UTVPI constraints.

In this paper, we examine proofs of infeasibility. In linear programs (systems of linear inequalities), we are interested in refutations that use the following inference rule:

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

Rule (2) is called the addition (ADD) rule and corresponds to the summation of constraints. The ADD rule plays a similar role in refutations of linear programs to the role played by resolution in refutations of CNF formulas. Observe that any assignment that satisfies the hypotheses of Rule (2) must also satisfy the consequent. Thus, Rule (2) is a sound inference rule.

Example (4): From the constraints $x_1 + x_2 \leq 3$ and $x_3 - x_2 \leq -1$, we can derive the constraint $x_1 + x_3 \leq 2$ by applying the ADD rule.

Additionally, if System (1) is unsatisfiable, then repeated applications of Rule (2) to the constraints in System (1) will result in a contradiction of the form: $0 \leq -a$, $a > 0$. Thus, Rule (2) is a complete inference rule.

The completeness of the ADD rule was established by Farkas (1902), in a lemma that is famously known as Farkas' Lemma for systems of linear inequalities (Schrijver 1987):

Lemma 1. *Let $\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$ denote a system of m linear constraints over n variables. Then, either $\exists \mathbf{x}\ \mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$ or (mutually exclusively), $\exists \mathbf{y} \in \Re_+^m\ \mathbf{y} \cdot \mathbf{A} = \mathbf{0},\ \mathbf{y} \cdot \mathbf{b} < 0$.*

In addition to Farkas' Lemma, there are additional lemmata that provide pairs of linear systems such that exactly one system in the pair is feasible. Such lemmata are called "Theorems of the Alternative" (Nemhauser and Wolsey 1999).

In this paper, we refer to the **y** variables in Farkas' Lemma as the Farkas' variables. These variables provide a certificate that proves that the original system is infeasible. In general, the Farkas' variables can assume any non-negative real value. However, we are interested only in cases where the Farkas' variables are restricted to the set {0, 1} (see Section 2.1).

For systems of UTVPI constraints, Rule (2) can be restricted as follows:

$$\frac{a_i \cdot x_i + a_j \cdot x_j \leq b_{k_1} \qquad -a_j \cdot x_j + a_l \cdot x_l \leq b_{k_2}}{a_i \cdot x_i + a_l \cdot x_l \leq b_{k_1} + b_{k_2}} \tag{3}$$

We refer to Rule (3) as the *transitive inference rule*. Although it is a restricted version of the ADD rule, it remains both sound and complete for the purposes of proving the linear infeasibility of UCSs (Lahiri and Musuvathi 2005).

https://doi.org/10.1017/S0960129523000300 Published online by Cambridge University Press

2.1 The read-once refutation (ROR) problem

We now define what it means for a refutation to be read-once.

Definition 4. *A refutation is said to be read-once, if each constraint is used at most once in the derivation of a contradiction.*

This restriction applies to both constraints in the original system as well as those derived from previous inferences. However, a derived constraint can be reused, if it can be rederived using a different set of input constraints.

Example (5): Consider the UCS defined by System (4).

$$\begin{aligned} l_1 &: \quad x_1 + x_2 \leq -2 \quad l_2 : -x_1 + x_2 \leq 0 \\ l_3 &: -x_2 + x_3 \leq 0 \quad\quad l_4 : -x_2 - x_3 \leq 1 \end{aligned} \tag{4}$$

System (4) has the following read-once refutation:

1. Apply the transitive inference rule to l_1 and l_2 to derive the constraint $l_5 : 2 \cdot x_2 \leq -2$.
2. Apply the transitive inference rule to l_3 and l_5 to derive the constraint $l_6 : x_2 + x_3 \leq -2$.
3. Apply the transitive inference rule to l_4 and l_6 to derive the contradiction $0 \leq -1$.

In this paper, we are interested in the problem of checking if a UCS has a read-once refutation.

Definition 5. *The read-once refutation (ROR) problem: Given a UCS* **U***, does* **U** *have a read-once refutation?*

Example (6): Consider the UCS defined by System (5).

$$\begin{aligned} l_1 &: \quad x_1 + x_2 \leq -2 \quad l_2 : -x_1 + x_4 \leq 1 \\ l_3 &: -x_1 - x_4 \leq 1 \quad\quad l_4 : -x_2 + x_3 \leq 0 \\ l_5 &: -x_2 - x_3 \leq 0 \end{aligned} \tag{5}$$

Observe that l_1 is the only constraint in System (5) with a negative defining constant. Thus, l_1 must be included in *any* refutation of System (5).

Any refutation of System (5) must derive a constraint of the form $0 \leq b$ where $b < 0$. Thus, all variables in l_1 must be eliminated by using other constraints. To eliminate x_1 from l_1, we must include either l_2 or l_3 in the refutation. However, if only one of these constraints is included, then the variable x_4 is not eliminated. Thus, both l_2 and l_3 must be in the refutation.

Similarly, to eliminate x_2 from l_1, we must include both l_4 and l_5. If both constraints are not used, then the variable x_3 is not eliminated.

Thus, any refutation of System (5) must include all five constraints in the system. However, the sum of these five constraints is the constraint $l_6 : -x_1 - x_2 \leq 0$. This is obviously not a contradiction. The only way to derive a contradiction is to include the constraint l_1 a second time. Thus, System (5) does not have a read-once refutation.

However, every infeasible UCS has a refutation in which each constraint is used at most twice (Subramani and Wojciechowski 2017).

In this paper, we study a variant of the ROR problem known as the CROR problem.

Definition 6. *The constraint-required ROR (CROR) problem in UCSs: Given a UCS* **U** *and a set of constraints* $S \subseteq$ **U***, does* **U** *have a read-once refutation that uses all of the constraints in S?*

https://doi.org/10.1017/S0960129523000300 Published online by Cambridge University Press

Given an unsatisfiable UCS **U**, the purpose of the ROR problem is to determine if **U** has a read-once refutation. In other words, we wish to find a subset of constraints in **U** whose sum is a contradiction of the form $0 \leq b$ where $b < 0$.

As stated previously, read-once refutations can be represented by placing a restriction on the Farkas' variables. Thus, the ROR problem can be modeled as an integer program. This integer program is represented by System 6.

$$\begin{aligned} \exists \mathbf{y}\, \mathbf{y} \cdot \mathbf{A} &= \mathbf{0} \\ \mathbf{y} \cdot \mathbf{b} &\leq -1 \\ \mathbf{y} &\in \{0,1\}^m \end{aligned} \tag{6}$$

Proposition 1. *Let R be a read-once refutation of a UCS* **U**. *If we add the constraints in R, we get a contradiction of the form:* $0 \leq b$, $b < 0$.

Proof. This follows immediately from System (6). □

2.2 The literal-once refutation (LOR) problem

We now define what it means for a refutation to be literal-once. First, we define a literal.

Definition 7. *In a UCS, a literal is either a variable* x_i *or its negation* $-x_i$.

Definition 8. *A refutation is said to be literal-once, if each literal is used at most once in the derivation of a contradiction.*

This restriction applies to both constraints in the original system as well as those derived from previous inferences. However, a derived constraint can be reused, if it can be rederived using a different set of input constraints.

Example (7): Consider the UCS defined by System (7).

$$\begin{aligned} l_1 &: \quad x_1 + x_2 \leq -2 \quad l_2 : -x_1 + x_4 \leq 0 \\ l_3 &: -x_2 + x_3 \leq 0 \quad\quad l_4 : -x_3 - x_4 \leq 1 \end{aligned} \tag{7}$$

System (7) has the following literal-once refutation:

1. Apply the transitive inference rule to l_1 and l_2 to derive the constraint $l_5 : x_2 + x_4 \leq -2$.
2. Apply the transitive inference rule to l_3 and l_5 to derive the constraint $l_6 : x_3 + x_4 \leq -2$.
3. Apply the transitive inference rule to l_4 and l_6 to derive the contradiction $0 \leq -1$.

In this paper, we are interested in the problem of checking if a UCS has a literal-once refutation.

Definition 9. *The literal-once refutation (LOR) problem: Given a UCS* **U**, *does* **U** *have a literal-once refutation?*

Example (8): Recall the UCS defined by System (4).

$$\begin{aligned} l_1 &: \quad x_1 + x_2 \leq -2 \quad l_2 : -x_1 + x_2 \leq 0 \\ l_3 &: -x_2 + x_3 \leq 0 \quad\quad l_4 : -x_2 - x_3 \leq 1 \end{aligned}$$

System (4) does not have a literal-once refutation. Observe that l_1 is the only constraint in System (4) with a negative right-hand side. Thus, l_1 must be in any refutation of System (4). Constraint l_1

https://doi.org/10.1017/S0960129523000300 Published online by Cambridge University Press

contains the literal x_1. Thus, any refutation of System (4) must use a constraint with the literal $-x_1$. The only such constraint is l_2. However, both l_1 and l_2 contain the literal x_2. Thus, any refutation of System (4) that uses both l_1 and l_2 is not a literal-once refutation. However, any refutation of System (4) must contain both constraints. This means that System (4) does not have a literal-once refutation.

In this paper, we study a variant of the LOR problem known as the CLOR problem.

Definition 10. *The constraint-required LOR (CLOR) problem in UCSs: Given a UCS* $\mathbf{U}$ *and a set of constraints* $S \subseteq \mathbf{U}$*, does* $\mathbf{U}$ *have a literal-once refutation that uses all of the constraints in S?*

Example (9): Consider the UCS defined by System (8).

$$\begin{aligned} l_1 &: x_1 + x_2 \leq -2 \quad l_2 : -x_1 - x_2 \leq 0 \\ l_3 &: x_1 + x_3 \leq 1 \quad\;\; l_4 : -x_1 - x_3 \leq 0 \end{aligned} \tag{8}$$

Let $S = \{l_1\}$. System (8) has the following CLOR that uses all the constraints in S:

1. Apply the transitive inference rule to l_1 and l_2 to derive the contradiction $0 \leq -2$.

Now let $S = \{l_3\}$. This constraint uses the literal x_3. The only constraint in S that uses the literal $-x_3$ is l_4. Thus, any refutation of System (8) that uses l_3 must also use l_4. Observe that the constraint l_3 uses the literal x_1 and the constraint l_4 uses the literal $-x_1$. This means that an LOR of System (8) that uses both l_3 and l_4 cannot use either l_1 or l_2. However, applying the transitive inference rule to l_3 and l_4 results in the constraint $0 \leq 1$. This is not a contradiction. Thus, System (8) does not have a CLOR that uses all the constraints in S.

2.3 The minimum weight perfect matching (MWPM) problem

The problem of finding a minimum weight perfect matching (MWPM) in an undirected, weighted graph is a well-known and well-studied problem (Cook et al. 1998). Our work in this paper focuses on establishing the equivalence of the MWPM problem and the CLOR and CROR problems in UCSs. Accordingly, we provide a brief overview of the MWPM problem.

Let $\mathbf{G} = \langle \mathbf{V}, \mathbf{E}, \mathbf{c} \rangle$ be an undirected graph, with vertex set $\mathbf{V}$, edge set $\mathbf{E}$, and edge cost function $\mathbf{c}$. A *matching* is any collection of vertex-disjoint edges. A *perfect matching* is a matching in which each vertex $v \in \mathbf{V}$ is matched. Without loss of generality, we assume that $|V|$ is even, since $\mathbf{G}$ cannot have a perfect matching, otherwise.

In this paper, we relate the ROR problem in UCSs to the decision version of the minimum weight perfect matching (MWPM_D) problem.

Definition 11. *The* $MWPM_D$ *problem: Given an undirected graph* $\mathbf{G} = \langle \mathbf{V}, \mathbf{E}, \mathbf{c} \rangle$ *and an integer L, does* $\mathbf{G}$ *have a perfect matching with weight at most L.*

Example (10): Let $\mathbf{G}$ be the graph in Fig. 1.

1. The edges $x_1 \overset{0}{\text{——}} x_2$, $x_4 \overset{-1}{\text{——}} x_6$, $x_5 \overset{1}{\text{——}} x_7$, $x_9 \overset{0}{\text{——}} x_{10}$, and $x_{11} \overset{0}{\text{——}} x_{12}$ form a matching of weight 0.
2. The edges $x_1 \overset{0}{\text{——}} x_2$, $x_3 \overset{0}{\text{——}} x_4$, $x_5 \overset{0}{\text{——}} x_6$, $x_7 \overset{0}{\text{——}} x_8$, $x_9 \overset{0}{\text{——}} x_{10}$, and $x_{11} \overset{0}{\text{——}} x_{12}$ form a perfect matching of weight 0.
3. The edges $x_2 \overset{-1}{\text{——}} x_3$, $x_4 \overset{-1}{\text{——}} x_6$, $x_5 \overset{1}{\text{——}} x_7$, $x_8 \overset{1}{\text{——}} x_9$, $x_{10} \overset{-1}{\text{——}} x_{11}$, and $x_{12} \overset{-1}{\text{——}} x_1$ form a minimum weight perfect matching of weight -2.

https://doi.org/10.1017/S0960129523000300 Published online by Cambridge University Press

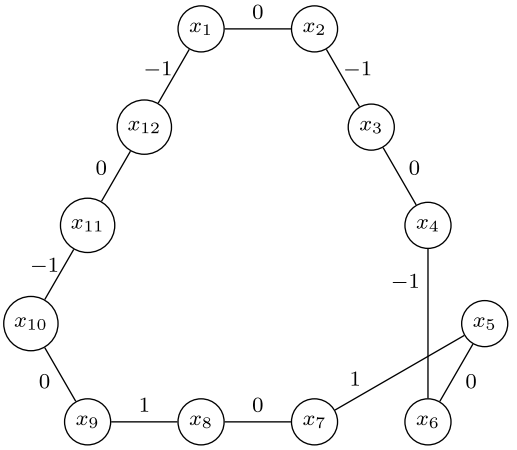


Figure 1. Undirected graph **G**.

The MWPM_D problem is one of the classical problems in combinatorial optimization (Korte and Vygen 2010). Over the years, there has been a steady stream of papers documenting improvements in algorithms for this problem (Duan et al. 2018; Edmonds 1967; Gabow 1976). While the MWPM_D problem is in **P**, it is unknown if the MWPM_D problem is in **NC**.

2.4 Complexity classes

We now define the complexity classes used in this paper. First, we define the complexity class **NC** (Papadimitriou 1994).

Definition 12. *A problem belongs to the class **NC**, if it can be solved in polylogarithmic parallel time using a polynomial number of processors.*

The class **NC** can be broken down further.

Definition 13. *A problem belongs to the class $\mathbf{NC}^i$, if it can be solved in $O(\log^i n)$ parallel time using a polynomial number of processors.*

We can also look at reductions that fall into these complexity classes.

Definition 14.
1. *An **NC** reduction can be performed in polylogarithmic parallel time using a polynomial number of processors.*
2. *An $\mathbf{NC}^i$ reduction is one that can be performed in $O(\log^i n)$ parallel time using a polynomial number of processors.*

Two problems which can be reduced to each other by **NC** reductions are known as **NC** equivalent.

For parallel algorithms, we have different ways to measure efficiency. These are known as work-optimality and work-efficiency (Khan et al. 2013).

Definition 15. *A parallel algorithm is work-optimal, if the total work done by the algorithm is within a constant factor of the work done by the best known sequential algorithm for the same problem.*

https://doi.org/10.1017/S0960129523000300 Published online by Cambridge University Press

Definition 16. *A parallel algorithm is work-efficient, if the total work done by the algorithm is within a logarithmic factor of the work done by the best known sequential algorithm for the same problem.*

3. Motivation and Related Work

Refutations are "no"-certificates, in that they provide an explanation for why a given constraint system is infeasible. Within the realm of Boolean formulas, resolution is one of the oldest and most widely used refutation techniques (Robinson 1965). Certificates enhance the reliability of responses provided by the implementation of an algorithm. For instance, consider an algorithm for checking Boolean satisfiability. Given an instance ϕ of a formula in Conjunctive Normal Form (CNF), a solver typically provides a "yes/no" answer. However, the answer itself does not serve as convincing evidence that the answer provided is correct. The solver could provide a satisfying assignment in case that it decides that ϕ is a "yes"-instance. This assignment serves as a certificate of feasibility and can be checked independently (Blum et al. 1990). The natural question is what form a certificate of infeasibility would take. In the case of clausal formulas, resolution refutations are the preferred form of negative certificates (Vinyals et al. 2018). In general, a resolution certificate of an arbitrary 3CNF formula is exponential in the size of the formula; indeed, exponential lower bounds on the size of resolution proofs were derived in Haken (1985). This is not surprising since, if every 3CNF formula had a short resolution refutation (or a short refutation in any reasonable proof system), then it would mean that **NP** = **coNP**. Since exponential length refutations are difficult to store and actually verify, research has proceeded along the lines of finding refutations in incomplete refutation systems. One such refutation system is the read-once refutation system. Read-once refutation, when specialized to resolution, is called read-once resolution (Iwama and Miyano 1995). Another line of research is to investigate the lengths of refutations and find the shortest refutation to explain the infeasibility of the formula. In Iwama (1997), it was shown that the problem of finding the shortest resolution proofs in arbitrary 3CNF formulas is **NP-complete**. A stronger result was obtained in Alekhnovich et al. (1998); they showed that the problem of finding the shortest resolution proof in Horn formulas is not linearly approximable, unless **P**=**NP**. This result is interesting because it is easy to see that every unsatisfiable Horn formula has a resolution refutation that is quadratic in the number of clauses.

In Iwama and Miyano (1995), it was shown that the ROR problem for resolution in arbitrary CNF formulas is **NP-complete**. This result was later strengthened by showing that the ROR problem for resolution in 3CNF formulas is **NP-complete** (Kleine Büning and Zhao 2002). In Szeider (2001), it was shown that the LOR problem for resolution in CNF formulas is **NP-complete**. It was later shown that the ROR problem for resolution in 2CNF formulas is **NP-complete** (Kleine Büning et al. 2018). In this paper, we examine these problems on continuous (as opposed to discrete) variables.

UTVPI constraints occur in a number of problem domains including but not limited to program verification (Lahiri and Musuvathi 2005), abstract interpretation (Cousot and Cousot 1977; Miné 2006), real-time scheduling (Gerber et al. 1995), and operations research (Hochbaum and (Seffi) Naor 1994). In particular, the Octagon Abstract Domain used in program verification is represented by UTVPI constraints (Harvey and Stuckey 1997; Miné 2006).

For systems of UTVPI constraints, the problem of checking for read-once refutations seems to be more difficult than that of checking for linear or even integer feasibility. Previous results have established that the problems of checking for linear feasibility and integer feasibility of a UCS can both be solved in $O(m \cdot n)$ time (Subramani and Wojciechowski 2017, 2018). However, checking for the existence of read-once refutations takes $O((m+n)^2 \cdot \log(m+n))$ time (Subramani and Wojciechowki 2019).

In Lahiri and Musuvathi (2005), the problem of checking the linear feasibility of a UCS was reduced to the problem of checking the linear feasibility of a system of difference constraints.

https://doi.org/10.1017/S0960129523000300 Published online by Cambridge University Press

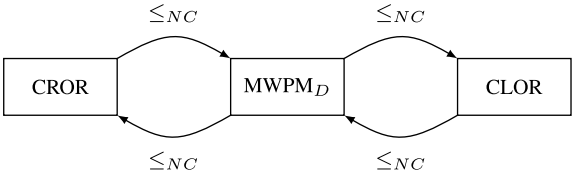


Figure 2. **NC** reductions.

Note that this problem is equivalent to the problem of finding shortest paths in a directed graph (Cormen et al. 2001) and thus belongs to the class **NC** (Leighton 1992).

In this paper, we show the equivalence of the CROR and CLOR problems in UCSs and the MWPM$_D$ problem. General matching problems have been extensively studied from the perspective of parallel complexity. However, it remains unknown if the MWPM$_D$ problem belongs to the complexity class **NC** (Anari and Vazirani 2020).

4. The CROR Problem in UTVPI Constraints

In this section, we show that the CROR problem in UTVPI constraints is **NC** equivalent to the MWPM$_D$ problem. This will be done by providing an **NC** reduction from the CROR problem to the MWPM$_D$ problem and an **NC** reduction from the MWPM$_D$ problem to the CROR problem. We will later do the same for the MWPM$_D$ problem and the CLOR problem. Fig. 2 shows these reductions.

First, we reduce the CROR problem to the MWPM$_D$ problem. This is done using a modified version of the reduction used in Subramani and Wojciechowki (2019). For the sake of completeness, we now describe that reduction.

Given a UCS $\mathbf{U} : \mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$, we construct the undirected graph $\mathbf{G} = \langle \mathbf{V}, \mathbf{E}, \mathbf{c} \rangle$ as follows:

1. For each variable x_i in $\mathbf{U}$, add the vertices x_i^+, $x_i'^+$, x_i^-, and $x_i'^-$ to $\mathbf{V}$. Additionally, add the edges $x_i^- \overset{0}{\text{——}} x_i^+$ and $x_i'^- \overset{0}{\text{——}} x_i'^+$ to $\mathbf{E}$.
2. Add the vertices x_0^+ and x_0^- to $\mathbf{V}$. Additionally, add the edge $x_0^- \overset{0}{\text{——}} x_0^+$ to $\mathbf{E}$.
3. For each constraint l_k of $\mathbf{U}$, add the vertices l_k and l_k' to $\mathbf{V}$ and the edge $l_k \overset{0}{\text{——}} l_k'$ to $\mathbf{E}$. Additionally:
   (a) If l_k is $x_i + x_j \leq b_k$, add the edges $x_i^+ \overset{\frac{b_k}{2}}{\text{——}} l_k$, $x_i'^+ \overset{\frac{b_k}{2}}{\text{——}} l_k$, $x_j^+ \overset{\frac{b_k}{2}}{\text{——}} l_k'$, and $x_j'^+ \overset{\frac{b_k}{2}}{\text{——}} l_k'$ to $\mathbf{E}$.
   (b) If l_k is $x_i - x_j \leq b_k$, add the edges $x_i^+ \overset{\frac{b_k}{2}}{\text{——}} l_k$, $x_i'^+ \overset{\frac{b_k}{2}}{\text{——}} l_k$, $x_j^- \overset{\frac{b_k}{2}}{\text{——}} l_k'$, and $x_j'^- \overset{\frac{b_k}{2}}{\text{——}} l_k'$ to $\mathbf{E}$.
   (c) If l_k is $-x_i + x_j \leq b_k$, add the edges $x_i^- \overset{\frac{b_k}{2}}{\text{——}} l_k$, $x_i'^- \overset{\frac{b_k}{2}}{\text{——}} l_k$, $x_j^+ \overset{\frac{b_k}{2}}{\text{——}} l_k'$, and $x_j'^+ \overset{\frac{b_k}{2}}{\text{——}} l_k'$ to $\mathbf{E}$.
   (d) If l_k is $-x_i - x_j \leq b_k$, add the edges $x_i^- \overset{\frac{b_k}{2}}{\text{——}} l_k$, $x_i'^- \overset{\frac{b_k}{2}}{\text{——}} l_k$, $x_j^- \overset{\frac{b_k}{2}}{\text{——}} l_k'$, and $x_j'^- \overset{\frac{b_k}{2}}{\text{——}} l_k'$ to $\mathbf{E}$.
   (e) If l_k is $x_i \leq b_k$, add the edges $x_i^+ \overset{\frac{b_k}{2}}{\text{——}} l_k$, $x_i'^+ \overset{\frac{b_k}{2}}{\text{——}} l_k$, $x_0^+ \overset{\frac{b_k}{2}}{\text{——}} l_k'$, and $x_0^- \overset{\frac{b_k}{2}}{\text{——}} l_k'$ to $\mathbf{E}$.
   (f) If l_k is $-x_i \leq b_k$, add the edges $x_i^- \overset{\frac{b_k}{2}}{\text{——}} l_k$, $x_i'^- \overset{\frac{b_k}{2}}{\text{——}} l_k$, $x_0^+ \overset{\frac{b_k}{2}}{\text{——}} l_k'$, and $x_0^- \overset{\frac{b_k}{2}}{\text{——}} l_k'$ to $\mathbf{E}$.

Observe that if $\mathbf{U}$ has m constraints over n variables, then $\mathbf{G}$ has $(4 \cdot n + 2 \cdot m + 2)$ vertices and $(2 \cdot n + 5 \cdot m + 1)$ edges. In other words, $\mathbf{G}$ has $O(m + n)$ vertices and $O(m + n)$ edges.

https://doi.org/10.1017/S0960129523000300 Published online by Cambridge University Press

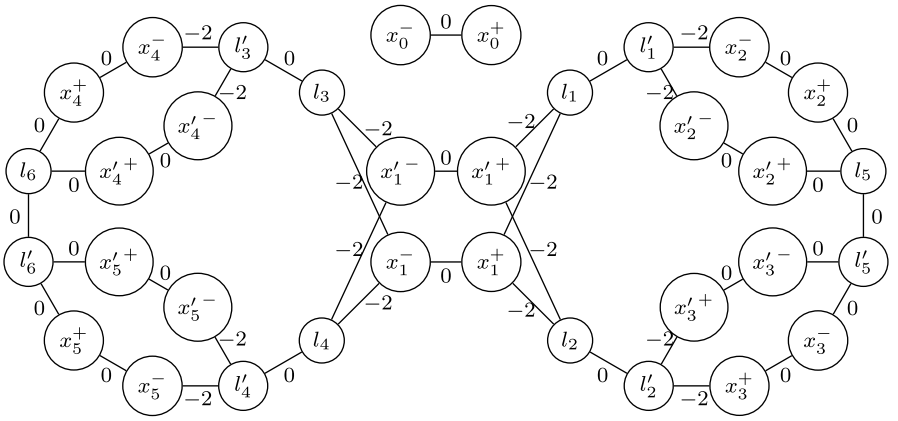


Figure 3. Undirected graph.

Example (11): Let us consider the UCS represented by System (9).

$$\begin{aligned} l_1 &: \quad x_1 - x_2 \leq -4 \quad l_2 : \quad x_1 + x_3 \leq -4 \\ l_3 &: -x_1 - x_4 \leq -4 \quad l_4 : -x_1 - x_5 \leq -4 \\ l_5 &: \quad x_2 - x_3 \leq 0 \quad l_6 : \quad x_4 + x_5 \leq 0 \end{aligned} \tag{9}$$

The undirected graph corresponding to UCS (9) is shown in Fig. 3.

The minimum weight perfect matching in this graph is $x_0^+ \overset{0}{\text{——}} x_0^-$, $x_1^+ \overset{-2}{\text{——}} l_2$, $l_2' \overset{-2}{\text{——}} x_3^+$, $x_3^- \overset{0}{\text{——}} l_5'$, $l_5 \overset{0}{\text{——}} x_2^+$, $x_2^- \overset{-2}{\text{——}} l_1'$, $l_1 \overset{-2}{\text{——}} x_1'^+$, $x_1'^- \overset{-2}{\text{——}} l_3$, $l_3' \overset{-2}{\text{——}} x_4^-$, $x_4^+ \overset{0}{\text{——}} l_6$, $l_6' \overset{0}{\text{——}} x_5^+$, $x_5^- \overset{-2}{\text{——}} l_4'$, $l_4 \overset{-2}{\text{——}} x_1^-$, $x_2'^+ \overset{0}{\text{——}} x_2'^-$, $x_3'^+ \overset{0}{\text{——}} x_3'^-$, $x_4'^+ \overset{0}{\text{——}} x_4'^-$, and $x_5'^+ \overset{0}{\text{——}} x_5'^-$. This matching has weight -16 and corresponds to the read-once refutation obtained by summing all six constraints. Note that this example is similar to Example 7 in Subramani and Wojciechowki (2019).

From Subramani and Wojciechowki (2019), **G** has a negative weight perfect matching, if and only if, **U** has a read-once refutation.

We now modify the reduction from Subramani and Wojciechowki (2019), to reduce the CROR problem in UCSs to the MWPM problem.

Let **U** be a UCS and let **G** be the corresponding undirected graph. If S is a set of constraints in **U**, let $\mathbf{G}'_S$ be the graph constructed by removing the edge $l_r \overset{0}{\text{——}} l_r'$ from **G** for each constraint $l_r \in S$.

We now relate the CROR problem in UCSs to the MWPM$_D$ problem.

Theorem 1. *Let* **U** *be a UCS and let S be a set of constraints in* **U**. **U** *has a read-once refutation that uses all of the constraints in S, if and only if,* $\mathbf{G}'_S$ *has a negative weight perfect matching.*

Proof. First, assume that **U** has a read-once refutation R that uses all of the constraints in S. Since R is a read-once refutation of **U**, the corresponding undirected graph **G** has a negative weight perfect matching M (Subramani and Wojciechowki 2019).

From Subramani and Wojciechowki (2019), for each constraint l_k in **U**, the perfect matching M uses the edge $l_k \overset{0}{\text{——}} l_k'$, if and only if, R does not use the constraint l_k. Let l_r be a constraint

https://doi.org/10.1017/S0960129523000300 Published online by Cambridge University Press

in S. Since R uses the constraint l_r, the edge $l_r \overset{0}{\text{——}} l'_r$ is not in M. Note that this is true for each constraint in S. Thus, M is a negative weight perfect matching of the graph $\mathbf{G}'_S$.

Now assume that $\mathbf{G}'_S$ has a negative weight perfect matching M. Note that M is also a negative weight perfect matching of $\mathbf{G}$. Thus, $\mathbf{U}$ has a read-once refutation R (Subramani and Wojciechowki 2019).

For each constraint l_k in $\mathbf{U}$, the perfect matching M uses the edge $l_k \overset{0}{\text{——}} l'_k$, if and only if, R does not use the constraint l_k. Let l_r be a constraint in S. Since M is a perfect matching of $\mathbf{G}'_S$, M does not use the edge $l_r \overset{0}{\text{——}} l'_r$. Thus, R uses the constraint l_r as desired. □

We now show that the graph $\mathbf{G}'_S$ can be constructed efficiently in parallel.

Theorem 2. *Given a UCS* $\mathbf{U}$ *with* m *constraints over* n *variables and a set* S *of constraints, the corresponding graph* $\mathbf{G}'_S$ *can be constructed in constant time using* $O(m+n)$ *processors.*

Proof. The construction of $\mathbf{G}'_S$ can be performed in parallel as follows:

1. For each $i = 1 \ldots n$, the i^{th} processor creates the vertices and edges corresponding to the variable x_i. These are the vertices and edges specified in step (1) of the construction of $\mathbf{G}$. All of these edges are also in $\mathbf{G}'_S$. Note that, in this step, each processor creates four vertices and two edges. Additionally, no two processors are required to access the same memory locations. Thus, this step can be performed in constant time in the CREW PRAM model.
2. For each $j = 1 \ldots m$, the j^{th} processor creates the vertices and edges corresponding to the constraint l_j. These are the vertices and edges specified in step (3) of the construction of $\mathbf{G}$. If $l_j \notin S$, then all of these edges are also in $\mathbf{G}'_S$. If $l_j \in S$, then the edge $l_j \overset{0}{\text{——}} l'_j$ is not in $\mathbf{G}'_S$. Note that, in this step, each processor creates two vertices and four or five edges. Additionally, no two processors are required to access the same memory locations. Thus, this step can be performed in constant time in the CREW PRAM model.

From this, it is easy to see that the reduction from the CROR problem in UCSs to the MWPM problem is an $\mathbf{NC}^0$ reduction. □

Note that this reduction works regardless of the size of S. Thus, the CROR problem can be $\mathbf{NC}$ reduced to the MWPM$_D$ problem even when $|S| \in O(m)$.

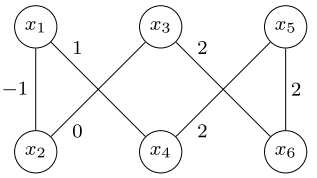
Now we need to show that this reduction can be performed in the opposite direction. That is, we want to construct an $\mathbf{NC}$ reduction from minimum weight perfect matching to the CROR problem in UCSs.

Let $\mathbf{G}$ be an undirected graph with n vertices and m edges, and let L be an arbitrary integer. From $\mathbf{G}$ and L, we construct a UCS $\mathbf{U}$ as follows.

1. For each vertex x_i in $\mathbf{G}$, create the variable x_i.
2. For each edge $x_i \overset{b_k}{\text{——}} x_j$ in $\mathbf{G}$, create the constraint $-x_i - x_j \leq b_k$.
3. Let $-C$ be the smallest weight of any edge in $\mathbf{G}$. If all edge weights are positive, then let $C = 0$. Additionally, let $W = \max\{m \cdot C + L + 1, 1\}$.
4. Create the constraint $x_1 \leq (n-1) \cdot W - L - 1$.
5. For each variable x_i, $i = 2, \ldots, n$, create the constraint $x_i \leq -W$.

Example (12): Consider the the undirected graph $\mathbf{G}$ and corresponding UCS $\mathbf{U}$, when $L = 4$, in Fig. 4.

https://doi.org/10.1017/S0960129523000300 Published online by Cambridge University Press



$$\begin{array}{rcl} -x_1 - x_2 & \leq & -1 \\ -x_2 - x_3 & \leq & 0 \\ -x_4 - x_5 & \leq & 2 \\ x_1 & \leq & 50 \\ x_3 & \leq & -11 \\ x_5 & \leq & -11 \end{array} \qquad \begin{array}{rcl} -x_1 - x_4 & \leq & 1 \\ -x_3 - x_6 & \leq & 2 \\ -x_5 - x_6 & \leq & 2 \\ x_2 & \leq & -11 \\ x_4 & \leq & -11 \\ x_6 & \leq & -11 \end{array}$$

Figure 4. Undirected graph and corresponding UCS.

Note that **U** has a read-once refutation R that uses the constraint $x_1 \leq 50$. R consists of the constraints $-x_1 - x_4 \leq 1$, $-x_2 - x_3 \leq 0$, $-x_5 - x_6 \leq 2$, $x_1 \leq 50$, and $x_2 \leq -11$ through $x_6 \leq -11$. Thus, **G** has a perfect matching of weight at most 4. This matching consists of the edges $x_1 \overset{1}{\text{———}} x_4$, $x_2 \overset{0}{\text{———}} x_3$, and $x_5 \overset{2}{\text{———}} x_6$.

We now show that **G** has a perfect matching of weight at most L, if and only if, **U** has a read-once refutation that uses the constraint $x_1 \leq (n-1) \cdot W - L - 1$.

Theorem 3. *Let* **G** *be an undirected graph and let L be an arbitrary integer,* **G** *has a perfect matching of weight at most L, if and only if, the corresponding UCS* **U** *has a read-once refutation that uses the constraint* $x_1 \leq (n-1) \cdot W - L - 1$.

Proof. Let M be a perfect matching in **G** with cost $c_M \leq L$. From M, we can construct a read-once refutation R of **U** as follows:

1. For each edge $x_i \overset{b_k}{\text{———}} x_j$ in M, add the constraint $-x_i - x_j \leq b_k$ to R.
2. Add the constraints $x_1 \leq (n-1) \cdot W - L - 1$, $x_2 \leq -W$, . . ., $x_n \leq -W$ to R.

Note that summing the constraints $x_1 \leq (n-1) \cdot W - L - 1$, $x_2 \leq -W$, . . ., $x_n \leq -W$ results in the constraint $\sum_{i=1}^{n} x_i \leq -L - 1$.

Since M is a perfect matching, every vertex in **G** is used by exactly one edge in M. Thus, each variable in **U** is used by exactly one constraint of the form $-x_i - x_j \leq b_k$ in R. Summing these constraints results in the constraint $-\sum_{i=1}^{n} x_i \leq c_M$. Summing this result with the constraint $\sum_{i=1}^{n} x_i \leq -L - 1$ results in the constraint $0 \leq c_M - L - 1$. Since $c_M \leq L$, this constraint is a contradiction. Thus, R is a refutation of **U**. Since R uses each constraint at most once, R is a read-once refutation as desired. Additionally, note that R uses each literal at most once. Thus, R is also an LOR.

Now assume that **U** has a read-once refutation R that uses the constraint $x_1 \leq (n-1) \cdot W - L - 1$. Any summation of the constraints in **U** corresponding to the edges in **G** results in a constraint with a defining constant H where $H \geq -C \cdot m$ since there are m such constraints and the defining constant of each constraint is at least $-C$. If R uses f constraints of the form $x_i \leq -W$, then summing the constraints in R would result in a constraint of the form $0 \leq (n-1-f) \cdot W - L - 1 + H$, where $H \geq -C \cdot m$.

Since R is a read-once refutation, $(n-1-f) \cdot W - L - 1 + H < 0$. Note that $W \geq C \cdot M + L + 1 \geq L + 1 - H$. Thus, $(n-1-f) \cdot W - L - 1 + H \geq (n-2-f) \cdot W$. Since, $(n-1-f) \cdot W - L - 1 + H < 0$ and $W \geq 1$, $(n-2-f) < 0$. Thus, $f \geq n-1$. Consequently, R must use all constraints of the form $x_i \leq -W$. Summing these constraints, together with the constraint $x_1 \leq (n-1) \cdot W - L - 1$, results in the constraint $\sum_{i=1}^{n} x_i \leq -L - 1$.

Since summing the constraints in R results in a contradiction of the form $0 \leq b$ where $b < 0$, the remaining constraints in R must sum together to produce a constraint of the form $-\sum_{i=1}^{n} x_i \leq c_M$ where $c_M \leq L$. By construction of **U**, each $-x_i$ term must come from a constraint of the form $-x_i - x_j \leq b_k$. Thus, the non-absolute constraints in R have the following properties:

https://doi.org/10.1017/S0960129523000300 Published online by Cambridge University Press

1. Each variable x_i is used by exactly one constraint in R of the form $-x_i - x_j \leq b_k$.
2. The defining constants of these constraints sum to the value $c_M \leq L$.

Thus, the edges corresponding to these constraints form a perfect matching in **G** with weight at most L. □

Now that we have established the correctness of the reduction from MWPM to the CROR problem in UCS, we need to show that this reduction is an **NC** reduction.

Theorem 4. *Given an undirected graph* **G** *with n vertices and m edges, the corresponding UCS* **U** *can be constructed in* $O(\log n)$ *time using* $O(m+n)$ *processors.*

Proof. The construction can be performed in parallel as follows:

1. Find C. Note that this can be done in $O(\log n)$ time using $O(n)$ processors using a divide and conquer parallel search procedure.
2. For each $j = 1 \ldots m$, the j^{th} processor creates the constraint corresponding to the j^{th} edge in **G**. This is the constraint specified in step (2) of the construction of **U**. Note that no two processors are required to access the same memory locations. Thus, this step can be performed in constant time in the CREW PRAM model.
3. For each $i = 2 \ldots n$, the i^{th} processor creates the constraint $x_i \leq -W$. Meanwhile, the first processor creates the constraint $x_1 \leq (n-1) \cdot W - L - 1$. Note that, in this step, no two processors are required to access the same memory locations. Thus, this step can be performed in constant time in the CREW PRAM model.

From this, it is easy to see that the reduction from the ROR problem in UCSs to minimum weight perfect matching can be accomplished by an $\mathbf{NC}^1$ reduction. □

Note that there is only one constraint that is required to be used by the read-once refutation of **U**. Thus, the MWPM_D problem can be **NC** reduced to the CROR problem in UCSs even when $|S| = 1$.

5. The CLOR Problem in UTVPI Constraints

In this section, we show that the CLOR problem in UTVPI constraints is **NC** equivalent to the MWPM_D problem.

First, we reduce the CLOR problem to the MWPM_D problem. This is done using a modified version of the reduction used in Subramani and Wojciechowki (2019). For the sake of completeness, we now describe that reduction.

Given a UCS $\mathbf{U} : \mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$, we construct the undirected graph $\mathbf{G} = \langle \mathbf{V}, \mathbf{E}, \mathbf{c} \rangle$ as follows:

1. Add the vertices x_0^+ and x_0^- to **V**. Additionally, add the edge $x_0^- \overset{0}{\text{——}} x_0^+$ to **E**.
2. For each variable x_i in **U**, add the vertices x_i^+ and x_i^- to **V**. Additionally, add the edge $x_i^- \overset{0}{\text{——}} x_i^+$ to **E**.
3. For each constraint l_k of **U**, add the vertices l_k and l'_k to **V** and the edge $l_k \overset{0}{\text{——}} l'_k$ to **E**. Additionally:
   (a) If l_k is of the form $x_i + x_j \leq b_k$, add the edges $x_i^+ \overset{\frac{b_k}{2}}{\text{——}} l_k$ and $x_j^+ \overset{\frac{b_k}{2}}{\text{——}} l'_k$ to **E**.
   (b) If l_k is of the form $x_i - x_j \leq b_k$, add the edges $x_i^+ \overset{\frac{b_k}{2}}{\text{——}} l_k$ and $x_j^- \overset{\frac{b_k}{2}}{\text{——}} l'_k$ to **E**.

https://doi.org/10.1017/S0960129523000300 Published online by Cambridge University Press

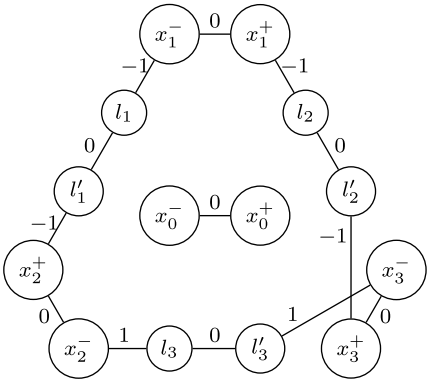


Figure 5. Undirected graph corresponding to UCS (10).

(c) If l_k is of the form $-x_i + x_j \leq b_k$, add the edges $x_i^- \xrightarrow{\frac{b_k}{2}} l_k$ and $x_j^+ \xrightarrow{\frac{b_k}{2}} l_k'$ to **E**.

(d) If l_k is of the form $-x_i - x_j \leq b_k$, add the edges $x_i^- \xrightarrow{\frac{b_k}{2}} l_k$ and $x_j^- \xrightarrow{\frac{b_k}{2}} l_k'$ to **E**.

(e) If l_k is of the form $x_i \leq b_k$, add the edges $x_i^+ \xrightarrow{\frac{b_k}{2}} l_k$, $x_0^+ \xrightarrow{\frac{b_k}{2}} l_k'$, and $x_0^- \xrightarrow{\frac{b_k}{2}} l_k'$ to **E**.

(f) If l_k is of the form $-x_i \leq b_k$, add the edges $x_i^- \xrightarrow{\frac{b_k}{2}} l_k$, $x_0^+ \xrightarrow{\frac{b_k}{2}} l_k'$, and $x_0^- \xrightarrow{\frac{b_k}{2}} l_k'$ to **E**.

Observe that if **U** has m constraints over n variables, then **G** has $(2 \cdot n + 2 \cdot m + 2)$ vertices and $(n + 3 \cdot m + N_a + 1)$ edges where N_a is the number of absolute constraints in **U**. Since $N_a \leq m$, **G** has $O(m + n)$ vertices and $O(m + n)$ edges.

Example (13): Let us consider the UCS represented by System (10).

$$l_1 : -x_1 + x_2 \leq -2 \quad l_2 : x_1 + x_3 \leq -2 \quad l_3 : -x_2 - x_3 \leq 2 \tag{10}$$

The undirected graph corresponding to UCS (10) is shown in Fig. 5.

The minimum weight perfect matching in this graph is $x_1^+ \xrightarrow{-1} l_2$, $l_2' \xrightarrow{-1} x_3^+$, $x_3^- \xrightarrow{1} l_3'$, $l_3 \xrightarrow{1} x_2^-$, $x_2^+ \xrightarrow{-1} l_1'$, and $l_1 \xrightarrow{-1} x_1^-$. This matching has weight -2 and corresponds to the literal-once refutation obtained by summing constraints l_1, l_2, and l_3. Note that this example is Example 5 in Subramani and Wojciechowki (2019).

From Subramani and Wojciechowki (2019), **G** has a negative weight perfect matching, if and only if, **U** has a literal-once refutation.

We now modify the reduction from Subramani and Wojciechowki (2019), to reduce the CLOR problem in UCSs to the MWPM problem.

Let **U** be a UCS and let **G** be the corresponding undirected graph. If S is a set of constraints in **U**, let $\mathbf{G}'_S$ be the graph constructed by removing the edge $l_r \xrightarrow{0} l_r'$ from **G** for each $l_r \in S$.

We now relate the CLOR problem in UCSs to the MWPM$_D$ problem.

Theorem 5. *Let* **U** *be a UCS and let S be a set of constraints in* **U**. **U** *has a literal-once refutation that uses all of the constraints in S, if and only if,* $\mathbf{G}'_S$ *has a negative weight perfect matching.*

Proof. First, assume that **U** has a literal-once refutation R that uses all of the constraints in S. Since R is a literal-once refutation of **U**, the corresponding undirected graph **G** has a negative weight perfect matching M (Subramani and Wojciechowki 2019).

https://doi.org/10.1017/S0960129523000300 Published online by Cambridge University Press

From Subramani and Wojciechowki (2019), for each constraint l_k in **U**, the perfect matching M uses the edge $l_k \overset{0}{\text{——}} l'_k$, if and only if, R does not use the constraint l_k. Let l_r be a constraint in S. Since R uses the constraint l_r, the edge $l_r \overset{0}{\text{——}} l'_r$ is not in M. Note that this applies to every constraint in S. Thus, M is a negative weight perfect matching of the graph $\mathbf{G}'_S$.

Now assume that $\mathbf{G}'_S$ has a negative weight perfect matching M. Note that M is also a negative weight perfect matching of **G**. Thus, **U** has a literal-once refutation R (Subramani and Wojciechowki 2019).

For each constraint l_k in **U**, the perfect matching M uses the edge $l_k \overset{0}{\text{——}} l'_k$, if and only if, R does not use the constraint l_k. Let l_r be a constraint in S. Since M is a perfect matching of $\mathbf{G}'_S$, M does not use the edge $l_r \overset{0}{\text{——}} l'_r$. Thus, R uses the constraint l_r as desired. □

Theorem 6. *Given a UCS* **U** *with m constraints over n variables, the corresponding graph* **G** *can be constructed in constant time using* $O(m + n)$ *processors.*

Proof. The construction can be performed in parallel as follows:

1. For each $i = 1 \ldots n$, the i^{th} processor creates the vertices and edges corresponding to the variable x_i. These are the vertices and edges specified in step (1) of the construction of **G**. All of these edges are also in $\mathbf{G}'_S$. Note that, in this step, each processor creates two vertices and one edge. Additionally, no two processors are required to access the same memory locations. Thus, this step can be performed in constant time in the CREW PRAM model.
2. For each $j = 1 \ldots m$, the j^{th} processor creates the vertices and edges corresponding to the constraint l_j. These are the vertices and edges specified in step (3) of the construction of **G**. If $l_j \notin S$, then all of these edges are also in $\mathbf{G}'_S$. If $l_j \in S$, then the edge $l_j \overset{0}{\text{——}} l'_j$ is not in $\mathbf{G}'_S$. Note that, in this step, each processor creates two vertices and two to four edges. Additionally, no two processors are required to access the same memory locations. Thus, this step can be performed in constant time in the CREW PRAM model.

From this, it is easy to see that the reduction from the CLOR problem in UCSs to the MWPM problem is an $\mathbf{NC}^0$ reduction. □

Now we need to show that this reduction can be performed in the opposite direction. That is, we want to construct an **NC** reduction from minimum weight perfect matching to the CLOR problem in UCSs.

By the proof of Theorem 3, the reduction from the MWPM_D problem to the CROR problem in UCSs is also a reduction to the CLOR problem in UCSs. Thus, that reduction is an **NC** reduction from minimum weight perfect matching to the CLOR problem in UCSs.

6. Conclusion

In this paper, we investigated the applicability of parallelization to the problem of finding CRORs and CLORs of systems of UTVPI constraints. In previous work (Subramani and Wojciechowki 2019), the ROR and LOR problems were reduced to the minimum weight perfect matching problem. We extended these results to a more restrictive form of refutation.

Additionally, we were able to reduce minimum weight perfect matching to the CROR and CLOR problems in UTVPI constraints. This establishes these problems as being equivalent to matching from the perspective of parallel algorithms. UTVPI constraints are an important class of linear constraints that find applications in abstract interpretation and program verification. It

https://doi.org/10.1017/S0960129523000300 Published online by Cambridge University Press

follows that certificates of infeasibility for UTVPI constraint systems are of enormous practical significance. Read-once and literal-once certificates are particularly useful in applications, since they are "short" by definition. We are currently looking into implementing this algorithm on a parallel computer in order to empirically validate our work.

From the perspective of future research, the following avenues appear promising:

1. What is the parallel complexity of the CROR problem in difference constraints? Note that difference constraints are a subset of UTVPI constraints. Thus, the CROR problem in difference constraints may be easier than the CROR problem in UTVPI constraints. This is supported by the fact that the reduction from MWPM to the CROR problem in UTVPI constraints utilizes constraints of the form $x_i + x_j \leq b$. Thus, the reduction in this paper does not work for difference constraints.
2. It is known that the minimum weight perfect matching problem for planar graphs belongs to the class **NC** (Anari and Vazirani 2017; Sankowski 2018). Can this result be extended to provide an **NC** algorithm for a restricted form of UTVPI constraint systems? For example, DCSs can be modeled using directed graphs. This means that we can consider DCSs that correspond to planar graphs. It may be possible to utilize the **NC** matching algorithm for planar graphs to develop an **NC** algorithm for this class of DCSs. We can then extend this result to a similar class of UCSs.
3. The problem of checking the linear feasibility of a UCS can be solved by an **NC** algorithm for shortest paths (Leighton 1992). Can a similar result be obtained for checking the integer feasibility of a UCS?

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

Competing interests. We declare that we have no competing interests.

References

Alekhnovich, M., Buss, S., Moran, S. and Pitassi, T. (1998). Minimum propositional proof length is **NP**-hard to linearly approximate. In: *Mathematical Foundations of Computer Science (MFCS)*, Lecture Notes in Computer Science, Springer-Verlag, 176–184.

Anari, N. and Vazirani, V. V. (2017). Planar graph perfect matching is in **NC**. In: *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, 650–661.

Anari, N. and Vazirani, V. V. (2020). Matching is as easy as the decision problem, in the **NC** model. In: *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12–14, 2020, Seattle, Washington, USA*, LIPIcs, vol. 151, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 54:1–54:25.

Blum, M., Luby, M. and Rubinfeld, R. (1990). Program result checking against adaptive programs and in cryptographic settings (extended abstract). In: *DIMACS Workshop on Distributed Computing and Crypthography*, American Mathematical Society, 107–118.

Cook, W., Cunningham, W. H., Pulleyblank, W. and Schrijver, A. (1998). *Combinatorial Optimization*, New York, NY, USA, John Wiley and Sons.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C. (2001). *Introduction to Algorithms*, Cambridge, MA, USA, MIT Press.

Cousot, P. and Cousot, R. (1977). Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: *POPL*, 238–252.

Duan, R., Pettie, S. and Su, H.-H. (2018). Scaling algorithms for weighted matching in general graphs. *ACM Transactions on Algorithms* **14** (1) 8:1–8:35.

Edmonds, J. (1967). An introduction to matching. Mimeographed notes. Engineering Summer Conference, University of Michigan, Ann Arbor, MI.

Farkas, G. (1902). Über die Theorie der Einfachen Ungleichungen. *Journal für die Reine und Angewandte Mathematik* **124** (124) 1–27.

Gabow, H. N. (1976). An efficient implementation of Edmonds' algorithm for maximum matching on graphs. *Journal of the ACM* **23** (2) 221–234.

https://doi.org/10.1017/S0960129523000300 Published online by Cambridge University Press

Gerber, R., Pugh, W. and Saksena, M. (1995). Parametric dispatching of hard real-time tasks. *IEEE Transactions on Computers* **44** (3) 471–479.

Haken, A. (1985). The intractability of resolution. *Theoretical Computer Science* **39** (2-3) 297–308.

Harvey, W. and Stuckey, P. J. (1997). A unit two variable per inequality integer constraint solver for constraint logic programming. In: *Proceedings of the 20th Australasian Computer Science Conference*, 102–111.

Hochbaum, D. S. and (Seffi) Naor, J. (1994). Simple and fast algorithms for linear and integer programs with two variables per inequality. *SIAM Journal on Computing* **23** (6) 1179–1192.

Iwama, K. (1997). Complexity of finding short resolution proofs. *Lecture Notes in Computer Science* **1295** 309–319.

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

Khan, M., Anisiu, M.-C., Domoszali, L., Iványi, A., Kasa, Z., Pirzada, S., Szécsi, L., Szidarovszky, F., Szirmay-Kalos, L. and Vizvári, B. (2013). *Algorithms of Informatics*, Budapest, Mondat Kft.

Kleine Büning, H., Wojciechowski, P. J. and Subramani, K. (2018). Finding read-once resolution refutations in systems of 2CNF clauses. *Theoretical Computer Science* **729** 42–56.

Kleine Büning, H. and Zhao, X. (2002). The complexity of read-once resolution. *Annals of Mathematics and Artificial Intelligence* **36** (4) 419–435.

Korte, B. and Vygen, J. (2010). *Combinatorial Optimization*, 4th edition, Algorithms and Combinatorics, vol. 21, New York, Springer-Verlag.

Lahiri, S. K. and Musuvathi, M. (2005). An efficient decision procedure for UTVPI constraints. In: *Proceedings of the 5th International Workshop on the Frontiers of Combining Systems, September 19–21, Vienna, Austria*, New York, Springer, 168–183.

Leighton, F. T. (1992). *Introduction to Parallel Algorithms and Architectures*, San Francisco, CA, USA, Morgan Kaufmann.

Miné, A. (2006). The octagon abstract domain. *Higher-Order and Symbolic Computation* **19** (1) 31–100.

Nemhauser, G. L. and Wolsey, L. A. (1999). *Integer and Combinatorial Optimization*, New York, John Wiley & Sons.

Papadimitriou, C. H. (1994). *Computational Complexity*, New York, Addison-Wesley.

Robinson, J. A. (1965). A machine-oriented logic based on the resolution principle. *Journal of the ACM* **12** (1) 23–41.

Sankowski, P. (2018). NC algorithms for weighted planar perfect matching and related problems. In: *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, Leibniz International Proceedings in Informatics (LIPIcs), vol. 107, Dagstuhl, Germany, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 97:1–97:16.

Schrijver, A. (1987). *Theory of Linear and Integer Programming*, New York, John Wiley and Sons.

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

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

Subramani, K. and Wojciechowski, P. J. (2018). A certifying algorithm for lattice point feasibility in a system of UTVPI constraints. *Journal of Combinatorial Optimization* **35** (2) 389–408.

Szeider, S. (2001). NP-completeness of refutability by literal-once resolution. In: *Automated Reasoning, First International Joint Conference, IJCAR 2001, Siena, Italy, June 18–23, 2001, Proceedings*, 168–181.

Vinyals, M., Elffers, J., Giráldez-Cru, J., Gocht, S. and Nordström, J. (2018). In between resolution and cutting planes: A study of proof systems for pseudo-boolean SAT solving. In: Beyersdorff, O. and Wintersteiger, C. M. (eds.) *Theory and Applications of Satisfiability Testing – SAT 2018*, Cham, Springer International Publishing, 292–310.

Cite this article: Subramani K and Wojciechowski P (2023). Constrained read-once refutations in UTVPI constraint systems: A parallel perspective. *Mathematical Structures in Computer Science*. https://doi.org/10.1017/S0960129523000300