

Incremental Satisfiability and Implication for UTVPI Constraints

Andreas Schutt and Peter J. Stuckey

NICTA Victoria Laboratory, Department of Computer Science & Software
Engineering, The University of Melbourne, Australia
{aschutt,pjs}@csse.unimelb.edu.au

Abstract. Unit two-variable-per-inequality (UTVPI) constraints form one of the largest class of integer constraints which are polynomial time solvable (unless $P=NP$). There is considerable interest in their use for constraint solving, abstract interpretation, spatial databases, and theorem proving. In this paper we develop a new incremental algorithm for UTVPI constraint satisfaction and implication checking that requires $\mathcal{O}(m + n \log n + p)$ time and $\mathcal{O}(n + m + p)$ space to incrementally check satisfiability of m UTVPI constraints on n variables and check implication of p UTVPI constraints.

1 Introduction

The unit two-variable-per-inequality (UTVPI) constraints form one of the largest class of integer constraints which are polynomial time solvable (unless $P=NP$). There is considerable interest in their use for constraint solving [7,6], abstract interpretation [9], spatial databases [11] and theorem proving [8]. In this paper we develop new incremental algorithms for UTVPI constraint satisfaction and implication.

A UTVPI constraint has the form $ax+by \leq d$ where x, y are integer variables, $d \in \mathbb{Z}$ and $a, b \in \{-1, 0, 1\}$. For example $x+y \leq 2$, $x-y \leq -1$, $0 \leq -1$ and $x \leq 2$ are UTVPI constraints. UTVPI constraint solving is based on transitive closure: A constraint $ax-y \leq d_1$ and $y+bz \leq d_2$ implies the constraint $ax+bz \leq d_1+d_2$. We can determine all the UTVPI consequences of a set of UTVPI constraints by transitive closure, but we need to *tighten* some constraints. The transitive closure procedure can generate constraints of the form $x+x \leq d$ and $-x-x \leq d$, which need to be tightened to $x \leq \lfloor \frac{d}{2} \rfloor$ and $-x \leq \lfloor \frac{d}{2} \rfloor$ respectively.

Jaffar et al. [7] and Harvey et al. [6] present incremental consistency checking algorithms for adding a UTVPI constraint c to a set ϕ of UTVPI constraints. They are based on maintaining the transitive and tight closure of the set of UTVPI constraints ϕ involving n variables. Both algorithms require $\mathcal{O}(n^2)$ time and $\mathcal{O}(n^2)$ space for an incremental satisfaction check. Both algorithms can also be used to incrementally check implication of UTVPI constraints by $\phi \cup \{c\}$. These algorithms require $\mathcal{O}(n^2 + p)$ time and $\mathcal{O}(n^2 + p)$ space for an incremental implication checking, where p is the number of constraints we need to check

for implication. In order to (non-incrementally) check satisfiability of m UTVPI constraints on n variables these approaches require $\mathcal{O}(n^2m)$ time, and to check implication they require $\mathcal{O}(n^2m + p)$ time.

An improvement on the complexity of (non-incremental) satisfiability for UTVPI constraints was devised by Lahiri and Musuvathi [8]. They define a non-incremental satisfiability algorithm requiring $\mathcal{O}(nm)$ time and $\mathcal{O}(n + m)$ space. The key behind their approach is to map UTVPI constraints to difference constraints (also called separation theory constraints) of the form $x - y \leq d$, where x and y are integer variables and $d \in \mathbb{Z}$.

The difference constraints are a well studied class of constraints because of their connection to shortest path problems. We can consider the constraint $x - y \leq d$ as a directed edge $x \rightarrow y$ with weight d . Satisfiability of difference constraints corresponds to the problem of negative weight cycle detection, and implication of difference constraints corresponds to finding shortest paths (see e.g. [3] for details).

The mapping of UTVPI to difference constraints by Lahiri and Musuvathi [8] is a relaxation of the problem. The relaxed problem is solved by a negative (weight) cycle detection algorithm but it guarantees only the satisfiability in \mathbb{Q} for the UTVPI problem. In order to check satisfiability in \mathbb{Z} they need to construct an auxiliary graph and check for certain paths in this graph.

In this paper we first extend Lahiri and Musuvathi's algorithm [8] to check satisfaction incrementally in $\mathcal{O}(n \log n + m)$ and $\mathcal{O}(n + m)$ space. Then we show how to build an incremental satisfiability and implication algorithm using the relaxation of Lahiri and Musuvathi and incremental approaches to implication for difference constraints of Cotton and Maler [3], which can incrementally check implication in $\mathcal{O}(n \log n + m + p)$ time and $\mathcal{O}(n + m + p)$ space.

2 Preliminaries

In this section we give notation and preliminary concepts.

A *weighted directed graph* $G = (V, E)$ is made up vertices V and a set E of weighted directed edges (u, v, d) from vertex $u \in V$ to vertex $v \in V$ with weight d . We also use the notation $u \xrightarrow{d} v$ to denote the edge (u, v, d) .

A *path* P from v_0 to v_k in graph G , denoted $v_0 \rightsquigarrow v_k$, is a sequence of edges e_1, \dots, e_k where $e_i = (v_{i-1}, v_i, d_i) \in E$. A *simple path* P is a path where $v_i \neq v_j, 0 \leq i < j \leq k$.

A (simple) *cycle* P is a path P where $v_0 = v_k$ and $v_i \neq v_j, 0 \leq i < j \wedge k \wedge (i \neq 0 \vee j \neq k)$.

The *path weight* of a path P , denoted $w(p)$ is $\sum_{i=1}^k d_i$.

Let G be a graph without negative weight cycles, that is without a cycle P where $w(P) < 0$. Then we can define the *shortest path* from v_0 to v_k , which we denote by $SP(v_0, v_k)$, as the (simple) path P from v_0 to v_k such that $w(P)$ is minimized.

Let $wSP(x, y) = w(SP(x, y))$ or $+\infty$ if no path exists from x to y .

Given a graph G and vertex x define the functions $\delta_x^{\leftarrow}, \delta_x^{\rightarrow} : V \rightarrow \mathbb{R}$ as

$$\delta_x^{\leftarrow}(y) = wSP(y, x) \quad \text{and} \quad \delta_x^{\rightarrow}(y) = wSP(x, y) .$$

Let G be a graph without negative weight cycles. Then π is a *valid potential function* for G if $\pi(u) + d - \pi(v) \geq 0$ for every edge (u, v, d) in G .

There are many algorithms (see e.g. [2]) for detecting negative weight cycles in a weighted directed graph, which either detect a cycle or determine a valid potential function for the graph.

Given a valid potential function π for graph $G = (V, E)$ we can define the *reduced cost graph* $rc(G)$ as $(V, \{(x, y, \pi(x) + d - \pi(y) \mid (x, y, d) \in E\})$. All weights in the reduced cost graph are non-negative and we can recover the original path length $w(P)$ for path P from x to y from paths in the reduced cost graph since $w(P) = w + \pi(y) - \pi(x)$ where w is the weight of the corresponding path in the reduced cost graph.

Since edges in the reduced cost graph are non-negative we can use Dijkstra's algorithm to calculate the shortest paths in the reduced cost graph in time $\mathcal{O}(n \log n + m)$ instead of $\mathcal{O}(nm)$.

2.1 Difference constraints

Difference constraints have the form $x - y \leq d$ where x and y are integer variables and $d \in \mathbb{Z}$. We can map difference constraints to a weighted directed graph.

Definition 1. Let C be a set of difference constraints and let $G = (V, E)$ be the graph comprised of one weighted edge $x \xrightarrow{d} y$ for every constraint $x - y \leq d$ in C . We call G the constraint graph of C .

The following well-known result characterizes how the constraint graph can be used for satisfiability and implication checking of difference constraints.

Theorem 1 ([3]). Let C be a set of difference constraints and G its corresponding graph. C is satisfiable iff G has no negative weight cycles, and if C is satisfiable then $C \models x - y \leq d$ iff $wSP(x, y) \leq d$.

2.2 UTVPI constraints

A UTVPI constraint is of the form $ax + by \leq d$, where x and y are integer variables, $a, b \in \{-1, 0, 1\}$, $c \in \{-1, 1\}$ and $d \in \mathbb{Z}$.

Definition 2. The transitive closure $TC(\phi)$ of a set of UTVPI constraints ϕ is defined as the smallest set S containing ϕ such that

$$ax - cy \leq d_1 \in S \wedge cy + bz \leq d_2 \in S \Rightarrow ax + bz \leq d_1 + d_2 \in S$$

The tightened closure $TI(\phi)$ of a set of UTVPI constraints ϕ is defined as the smallest set S containing ϕ such that

$$ax + ax \leq d \in S \Rightarrow ax \leq \left\lfloor \frac{d}{2} \right\rfloor \in S, \quad a \in \{-1, 1\}$$

Table 1. Transformation from UTVPI constraint c to associated difference constraints $D(c)$ to edges in the constraint graph $E(c)$.

UTVPI c	Diff. Constr. $D(c)$	Edges $E(c)$
$x - y \leq d$	$x^+ - y^+ \leq d$ $y^- - x^- \leq d$	$y^+ \xrightarrow{d} x^+$ $x^- \xrightarrow{d} y^-$
$x + y \leq d$	$x^+ - y^- \leq d$ $y^+ - x^- \leq d$	$y^- \xrightarrow{d} x^+$ $x^- \xrightarrow{d} y^+$
$-x - y \leq d$	$x^- - y^+ \leq d$ $y^- - x^+ \leq d$	$y^+ \xrightarrow{d} x^-$ $x^+ \xrightarrow{d} y^-$
$x \leq d$	$x^+ - x^- \leq 2d$	$x^- \xrightarrow{2d} x^+$
$-x \leq d$	$x^- - x^+ \leq 2d$	$x^+ \xrightarrow{2d} x^-$

The tightened transitive closure $TTC(\phi)$ of ϕ is the smallest set containing ϕ that satisfies both conditions.

The fundamental result for UTVPI constraints solving is:

Theorem 2 ([7]). *Let ϕ be a set of UTVPI constraints. Then ϕ is unsatisfiable iff exists $0 \leq d \in TTC(\phi)$ where $d < 0$.*

We can extend this for implication checking straightforwardly:

Corollary 1. *Let ϕ be a satisfiable set of UTVPI constraints. Then $\phi \models ax + by \leq d$ iff $ax + by \leq d' \in TTC(\phi)$ with $d' \leq d$ or $\{ax \leq d_1, by \leq d_2\} \subseteq TTC(\phi)$ with $d_1 + d_2 \leq d$.*

Example 1. Consider the UTVPI constraints $\phi \equiv \{x - y \leq 2, x + y \leq -1, -x - z \leq -4\}$, Then $TC(\phi)$ includes in addition $\{x + x \leq 1, -y - z \leq -2, y - z \leq -5, -z - z \leq -7, x - z \leq -3\}$. And $TI(TC(\phi))$ includes in addition $\{x \leq 0, -z \leq -4\}$ and $TTC(\phi) = TI(TC(\phi))$ in this case. The constraint $-z \leq -3$ is implied by ϕ as is $y - z \leq 0$.

3 Lahiri and Musuvathi's approach

Lahiri and Musuvathi map UTVPI constraints ϕ to difference constraints or equivalently a weighted directed graph G_ϕ , and they use graph algorithms to detect satisfiability.

We denote the constraint graph arising from ϕ as $G_\phi = (V, E)$. The graph G contains two vertices x^+ and x^- for every variable x . These variables are used to convert UTVPI constraints into difference constraints. The vertex x^+ represents $+x$ and x^- represents $-x$.

Let ϕ be a set of UTVPI constraints. Each UTVPI constraint $c \in \phi$ is mapped to a set of difference constraints $D(c)$, or equivalently a set of weighted edges $E(c)$. The mapping is shown in the Table 1. Each UTVPI constraint on two

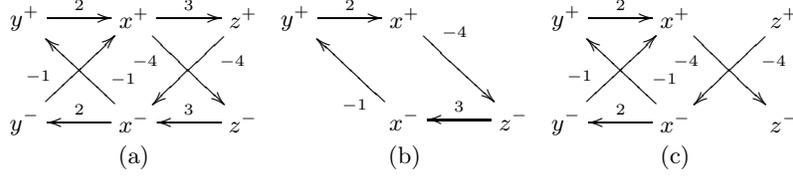


Fig. 1. (a) $G_{\phi'}$ for ϕ' of Example 2 which is \mathbb{Q} feasible but not \mathbb{Z} feasible. (b) a zero length cycle in $G_{\phi'}$. (c) G_ϕ for ϕ of Example 1.

variables generates two difference constraints and accordingly two edges in the constraint graph. Each UTVPI constraint on a single variable generates a single constraint, and hence a single edge.

Let $-v$ denote the counterpart of a vertex $v \in V$, i.e. $-x^+ := x^-$ and $-x^- := x^+$. Clearly, for each edge $(x, y, d) \in E$ the graph G_ϕ also includes the edge $(-y, -x, d)$ with equal weight. This correspondence extends to paths.

Lemma 1 ([8]). *If there is a path P from u to v in the constraint graph G_ϕ , then there is a path P' from $-v$ to $-u$ such that $w(P) = w(P')$.*

If we relax the restriction on variables to take values in \mathbb{Z} and allow them to take values in \mathbb{Q} we can check satisfiability in \mathbb{Q} using G_ϕ .

Lemma 2 ([8]). *A set of UTVPI constraints ϕ is unsatisfiable in \mathbb{Q} if and only if the constraint graph $G_\phi = (V, E)$ contains a negative weight cycle.*

The reason why the feasibility in \mathbb{Z} cannot be tested with G_ϕ arises from the possible implication of constraints of the form $x + x \leq d$ or $-x - x \leq d$ through the transitivity of constraints in ϕ . If d is odd (equivalently $d/2 \in \mathbb{Q} \setminus \mathbb{Z}$) then ϕ may be satisfiable with $x = d/2$ but not with $x = \lfloor d/2 \rfloor$.

Example 2. Consider the UTVPI problem $\phi' \equiv \{x - y \leq 2, x + y \leq -1, -x - z \leq -4, -x + z \leq 3\}$, then a transitive consequence of the first two is $x + x \leq 1$, while a consequence of the second two is $-x - x \leq -1$. Together these require $x = \frac{1}{2}$.

The graph $G_{\phi'}$ is shown in Figure 1(a). A zero length cycle is extracted in Figure 1(b). This cycle has solutions in \mathbb{Q} but not in \mathbb{Z} . \square

The satisfiability algorithm of Lahiri and Musuvathi [8] is based on Lemma 2 and the following result.

Lemma 3 ([8]). *Suppose G_ϕ has no negative cycles and ϕ is unsatisfiable in \mathbb{Z} . Then G_ϕ contains a zero weight cycle containing vertices u and $-u$ such that $wSP(u, -u)$ is odd.*

The algorithm first checks \mathbb{Q} feasibility using a negative cycle detection algorithm, and then checks that no such zero weight cycles exists in G_ϕ .

Algorithm 1: LAMU

Input: ϕ a set of UTVPI constraints
Output: SAT if ϕ is satisfiable, UNSAT otherwise

- 1 Construct the constraint graph $G_\phi = (V, E)$ from ϕ ;
- 2 Run a negative cycle detection algorithm on G_ϕ ;
- 3 **if** G_ϕ contains a negative cycle **then**
- 4 **return** UNSAT
- 5 **else**
- 6 let π be a valid potential function for G_ϕ
- 7 $E' := \{(u, v) \mid (u, v, d) \in E, \pi(u) + d = \pi(v)\}$;
- 8 $G'_\phi := (V, E')$;
- 9 Group the vertices in G'_ϕ into strongly connected components (SCCs). Vertices u and v are in the same SCC if and only if there is a path from u to v and a path from v to u in G'_ϕ . u and v are in the same SCC exactly when there is a zero-weight cycle in G_ϕ containing u and v ;
- 10 **for all** $u \in V$ **do**
- 11 **if** $-u$ is in the same SCC as u **and** $\pi(-u) - \pi(u)$ is odd **then**
- 12 **return** UNSAT
- 13 **return** SAT

Example 3. A valid potential function for the graph shown in Figure 1(a) is $\pi(y^+) = 0$, $\pi(x^+) = 2$, $\pi(z^+) = 5$, $\pi(y^-) = 3$, $\pi(x^-) = 1$, $\pi(z^-) = -2$. Each of the arcs is tight, so E' contains all edges, and all nodes are in the same SCC. Both x^+ and x^- occur in the same SCC and $SP(x^+, x^-) = \pi(x^-) - \pi(x^+) = -1$ is odd, hence the system is unsatisfiable.

The complexity is $\mathcal{O}(nm)$ time and $\mathcal{O}(n + m)$ space assuming we use a Bellman-Ford single source shortest path algorithm [1,4] for negative cycle detection.

4 Incremental UTVPI Satisfaction

The incremental satisfiability problem is: Given a satisfiable set of UTVPI ϕ (with n variables and m constraints) and UTVPI constraint c , determine if $\phi \cup \{c\}$ is satisfiable. In this section we define an incremental satisfiability checker for UTVPI constraints that requires $\mathcal{O}(n \log n + m)$ time and $\mathcal{O}(n + m)$ space. It relies on simply making incremental the algorithm LAMU of Lahiri and Musuvathi.

The key is to incrementalize the negative cycle detection. We use an algorithm due to Frigioni *et al.* [5], using the simplified form (Algorithm 2: INCCONDIFF) of Cotton and Maler [3] (since we are not interested in edge deletion). Given a graph $G = (V, E)$ and valid potential function π for G and edge $e = u \xrightarrow{d} v$, this algorithm returns $G' = (V, E \cup \{e\})$ and a valid potential function π' for G' or determines a negative cycle and returns UNSAT. The complexity is $\mathcal{O}(n \log n + m)$ time and $\mathcal{O}(n + m)$ space using Fibonacci heaps to implement argmin.

Algorithm 2: INCCONDIFF

Input: $G_\phi = (V, E)$ a graph, π a valid potential function for G_ϕ , edge (u, v, d) a new constraint to add to G_ϕ .

Output: UNSAT if $\phi \cup \{u - v \leq d\}$ is unsatisfiable, or $G_{\phi \cup \{u - v \leq d\}}$ and a valid potential function π' for $G_{\phi \cup \{u - v \leq d\}}$.

- 1 $\gamma(v) := \pi(u) + d = \pi(v)$;
- 2 $\gamma(w) := 0$ for all $w \neq v$;
- 3 **while** $\min(\gamma) < 0 \wedge \gamma(u) = 0$ **do**
- 4 $s := \operatorname{argmin}(\gamma)$;
- 5 $\pi'(s) := \pi(s) + \gamma(s)$;
- 6 $\gamma(s) := 0$;
- 7 **for all** $s \xrightarrow{d'} t \in G$ **do**
- 8 **if** $\pi'(t) = \pi(t)$ **then**
- 9 $\gamma(t) := \min\{\gamma(t), \pi'(s) + d' - \pi'(t)\}$
- 10 **if** $\gamma(u) < 0$ **then**
- 11 **return** UNSAT
- 12 **return** $((V, E \cup \{(u, v, d)\}), \pi')$

The incremental UTVPI satisfiability algorithm simply runs INCCONDIFF at step 2 of LAMU, the remainder of the algorithm is unchanged. Since the remainder of the the LAMU algorithm requires $\mathcal{O}(n + m)$ time and space, the complexity bounds are the same as for the incremental negative cycle detection algorithm.

5 Incremental UTVPI Implication

The incremental implication problem is given a set P of p UTVPI constraints and a satisfiable set ϕ of m UTVPI constraints on n variables, where $\phi \not\models c', \forall c' \in P$, as well as a single new UTVPI constraint c , check for each $c' \in P$ if $\phi \wedge c \models c'$.

Incremental implication is important if we wish to use UTPVI constraints in a Satisfiability Modulo Theories (SMT) solver [10], as well as for uses in abstract interpretation and spatial databases. Our approach to incremental implication is similar to the approach of Cotton and Maler [3] for incremental implication for difference constraints.

The key to the algorithm are the following three results.

Lemma 4. *Let $ax + by \leq d \in TTC(\phi)$ where $\{a, b\} \subseteq \{-1, 1\}$, then $ax + by \leq d \in TC(\phi)$.*

The result holds since tightening introduces constraints involving a single variable and any further transitive closure involving them can only create new constraints involving a single variable.

Lemma 5. *Let $ax \leq d \in TTC(\phi)$ where $a \in \{-1, 1\}$ then $ax \leq d \in TI(TC(\phi))$.*

The result holds since any result of transitive closure on a new UTVPI constraint $by \leq d'$ introduced by tightening, can be mimicked using the constraint $by + by \leq \{2d', 2d' + 1\}$ that introduced it, and tightening the end result.

The above two results show that $TC(\phi)$ is the crucial set of interest for UTVPI implication checking. The following result shows how we can use the constraint graph to reason about $TC(\phi)$.

Lemma 6. $c \in TC(\phi)$ iff there is a cycle of length d , in the case of $c \equiv 0 \leq d$, or a path $u \rightsquigarrow v$ of length d in G_ϕ where $(u, v, d) \in E(c)$.

Proof. This lemma follows straightforward of the definition of $TC(\phi)$ and the transformation of the set of UTVPI constraints to its constraint graph. It can be prove easily by induction over the number of transitive closure steps in $TC(\phi)$ resp. the length of cycle or path in G_ϕ . \square

Example 4. Consider ϕ of Example 1. Then for example $x + x \leq 1 \in TC(\phi)$ and there is a path $x^- \rightsquigarrow x^+$ of length 1 in G_ϕ shown in Figure 1(c). Similarly $y - z \leq -5 \in TC(\phi)$ and there are paths $y^- \rightsquigarrow z^-$ and $z^+ \rightsquigarrow y^+$ of length -5 in G_ϕ .

We can use paths (in particular shortest paths) in G_ϕ to reason about most constraints in $TTC(\phi)$. In order to handle tightening we introduce a *bounds function* ρ which records the upper and lower bounds for each variable x , on the vertices x^+ and x^- . It is defined as:

$$\rho(u) = \left\lfloor \frac{wSP(u, -u)}{2} \right\rfloor.$$

We can show that $\rho(x^-)$ computes the the upper bound of x and $-\rho(x^+)$ is the lower bound of x . Using Lemmas 5 and 6 we have.

Lemma 7. For UTVPI constraints ϕ ,

$$\begin{aligned} \rho(x^-) &= \min\{d \mid x \leq d \in TTC(\phi)\} \\ \rho(x^+) &= \min\{d \mid -x \leq d \in TTC(\phi)\} \end{aligned}$$

where we assume $\min \emptyset = +\infty$.

Example 5. Consider the graph in Figure 1(c) for constraints ϕ of Example 1. Then $\rho(x^-) = 0$ since $wSP(x^-, x^+)$ equals to 1 and $x \leq 0 \in TTC(\phi)$, while $\rho(z^+) = -4$ since $wSP(z^+, z^-) = -7$ and $-z \leq -4 \in TTC(\phi)$. Note e.g. $\rho(x^+) = +\infty$ and there is no constraint of the form $-x \leq d$ in $TTC(\phi)$.

The key to incremental satisfaction is the following result.

Theorem 3. If the constraint graph G_ϕ contains no negative weight cycle (i.e. ϕ is satisfiable in \mathbb{Q}) then ϕ is unsatisfiable in \mathbb{Z} iff a vertex $v \in V$ exists with $\rho(v) + \rho(-v) < 0$.

Proof. Let ϕ be a satisfiable set of UTVPI constraints in \mathbb{Q} . Because of the Lemma 6 it applies the non-existence of a constraint $0 < d \in TC(\phi)$ where $d < 0$. Therefore ϕ is unsatisfiable in \mathbb{Z} iff such a constraint belongs to $TTC(\phi) \setminus TC(\phi)$ (Theorem 2), i.e. a possible unsatisfiability is caused by tightening.

The Lemma 4 implies the equivalence for each constraint $c \in TTC(\phi) \setminus TC(\phi)$ to $ax \leq d$ where $a \in \{-1, 0, 1\}$. Hence, ϕ is unsatisfiable in \mathbb{Z} iff two constraints $x \leq d_1$ and $-x \leq d_2$ with $d_1 + d_2 < 0$ exist in $TTC(\phi)$ iff (Lemma 7) $\rho(x^+) + \rho(x^-) < 0$. \square

Effectively failure can only be caused by tightening if the bounds of a single variable contradict.

Example 6. Consider the graph in Figure 1(a) for constraints ϕ' of Example 2. There is no negative weight cycle in $G_{\phi'}$ but $\rho(x^-) = 0$ and $\rho(x^+) = -1$ because of $x^+ \xrightarrow{-4} z^- \xrightarrow{-3} x^-$. Hence the system is unsatisfiable.

Similarly the key to incremental implication is the following rephrasing of Corollary 1.

Theorem 4. *If ϕ is a satisfiable set of UTVPI constraints then $\phi \models c$ iff for all $(u, v, d) \in E(c)$ either $wSP(u, v) \leq d$ or $\rho(u) + \rho(-v) \leq d$.*

Proof. Let ϕ be a satisfiable set of UTVPI constraints. Because of Corollary 1 it holds $\phi \models c$ and $c \equiv ax + by \leq d$ iff $ax + by \leq d' \in TTC(\phi)$ and $d' \leq d$ or $\{ax \leq d_1, by \leq d_2\} \subseteq TTC(\phi)$ and $d_1 + d_2 \leq d$.

Now, the theorem holds straightforward due to Lemma 7 for the constraints with one variable, and Lemma 4 and 6 for the other constraints. \square

Example 7. Consider the graph in Figure 1(c) for constraints ϕ of Example 1. $\phi \models -z \leq -3$ is shown since $wSP(z^+, z^-) = -7 \leq 2 \times -3$.

Algorithm 3 shows the new algorithm. As input it takes the constraint graph G_ϕ , a valid potential function π , the bounds function ρ , a set P of UTVPI constraints to check for implication, as well as the UTVPI constraint c which should be added to ϕ .

In the first step (line 1) the constraint c is transformed to its corresponding edges $E(c)$ in a constraint graph. Then each edge in $E(c)$ is added consecutively to the constraint graph G_ϕ by using the INCCONDIFF algorithm of Cotton and Maler [3]. After inserting all edges in G' , the constraint graph equals to $G_{\phi \cup \{c\}}$ and π' is its valid potential function for G' . Hence $\phi \cup \{c\}$ is satisfiable in \mathbb{Q} . The remainder of the algorithm maintains the bounds function ρ' (lines from 8 to 12) and it is used to test the feasibility in \mathbb{Z} (lines 13 and 14), and the implication of constraints in P (lines 15 to 18).

By Lemma 6 to maintain ρ we need to see if the shortest path from x to $-x$ has changed. We only need to scan for new shortest paths using the newly added edges. We can restrict attention to a single added edge (u, v, d) since if there is a path from x over the edge (u, v, d) to $-x$ ($x^+ \rightsquigarrow u \xrightarrow{d} v \rightsquigarrow x^-$) then because of

Algorithm 3: SCST – Incremental satisfiability and implication for UTVPI constraints.

Input: $G_\phi = (V, E)$ a *constraint graph* representing set of UTVPI constraints ϕ , π a *valid potential function* on G_ϕ , ρ the *bound function* of ϕ , P a set of UTVPI constraints not implied by ϕ , and a UTVPI constraint c to be added.

Output: $G_{\phi \cup \{c\}}$, its *valid potential function* π' and the bound function ρ' of $\phi \cup \{c\}$ and the set $P' \subseteq P$ of constraints not implied by $\phi \cup \{c\}$, or UNSAT if $\phi \cup \{c\}$ is not satisfiable.

- 1 $G' := G_\phi$, $\pi' := \pi$, $\rho' = \rho$, compute $E(c)$;
- 2 **for all** $e \in E(c)$ **do**
- 3 $res := \text{INCCONDIFF}(G', \pi', e)$;
- 4 **if** $res = \text{UNSAT}$ **then**
- 5 **return** UNSAT
- 6 **else**
- 7 $(G', \pi') := res$
- 8 let (u, v, d) be any edge in $E(c)$;
- 9 compute δ_u^{\leftarrow} and δ_v^{\rightarrow} by using the reduced cost graph for G' via π' ;
- 10 **for all** $x \in V$ **do**
- 11 $sp := \delta_u^{\leftarrow}(x) + d + \delta_v^{\rightarrow}(-x)$;
- 12 $\rho'(x) := \min\{\rho(x), \lfloor \frac{sp}{2} \rfloor\}$;
- 13 **for all** $x \in V$ **do**
- 14 **if** $\rho'(x) + \rho'(-x) < 0$ **then return** UNSAT;
- 15 $P' := \emptyset$;
- 16 **for all** $c' \in P$ **do**
- 17 $(x, y, d') := \text{first element in } E(c')$;
- 18 **if** $\delta_u^{\leftarrow}(x) + d + \delta_v^{\rightarrow}(y) > d'$ **and** $\delta_u^{\leftarrow}(-y) + d + \delta_v^{\rightarrow}(-x) > d'$ **and** $\rho'(x) + \rho'(-y) > d'$ **then** $P' := P' \cup \{c'\}$;
- 19 **return** (G', π', ρ', P')

Lemma 1 there is equal-weight path from x via the “counter-edge” $(-v, -u, d)$ to $-x$ ($x^+ \equiv -x^- \rightsquigarrow -v \xrightarrow{d} -u \rightsquigarrow -x^+ \equiv x^-$).

We calculate the shortest paths in $G_{\phi \cup \{c\}}$ from each vertex x to u ($\delta_u^{\leftarrow}(x)$) and from v to each vertex x ($\delta_v^{\rightarrow}(x)$) (line 9). The shortest path for δ_u^{\leftarrow} can be computed like δ_u^{\rightarrow} by simply reversing the edges in the graph.

We can then calculate the shortest path from x to $-x$ via the edge $u \xrightarrow{d} v$ using the path $x^+ \rightsquigarrow u \xrightarrow{d} v \rightsquigarrow x^-$ as $\delta_u^{\leftarrow}(x) + d + \delta_v^{\rightarrow}(-x)$. We update ρ' if required (line 12).

We can now check satisfiability of $\phi \cup \{c\}$ in \mathbb{Z} using Theorem 3 (lines 13 and 14). Finally we check implications using Theorem 4.

Using the above results, it is not difficult to show that the algorithm is correct with the desired complexity bounds.

Theorem 5. *Algorithm 3 (SCST) is correct and runs in $\mathcal{O}(n \log n + m + p)$ time and $\mathcal{O}(n + m + p)$ space.*

Proof. The algorithm is correct if it returns UNSAT in the case of unsatisfiability of $\phi \cup \{c\}$ or the constraint graph $G_{\phi \cup \{c\}}$, its valid potential function π' , its bounds function ρ' and the set of constraints $P' \subseteq P$ not implied by $\phi \cup \{c\}$.

The Lemma 2 and the Algorithm INCCONDIFF (see Cotton and Maler [3]) guarantee that after termination of INCCONDIFF $G' = G_{\phi \cup \{c\}}$ and π' is its valid potential function if $\phi \cup \{c\}$ is satisfiable in \mathbb{Q} ; otherwise $\phi \cup \{c\}$ is unsatisfiable and the algorithm returns UNSAT.

After application of INCCONDIFF the algorithm maintains the bounds function (lines 8 to 12) by calculation of the shortest path $x \rightsquigarrow u \rightarrow v \rightsquigarrow -x$ via one added edge $(u, v, d) \in E(c)$ for each node x in $G_{\phi \cup \{c\}}$. Remark: we only have to consider the shortest paths via the added edges ρ give us the length of a shortest path without those added edges. Due to Theorem 3 the algorithm checks $\phi \cup \{c\}$ for unsatisfiability in \mathbb{Z} in the next two lines. If it is unsatisfiable ScSt terminates and returns UNSAT.

The remainder of the algorithm computes the set of non-implied constraints $P' \subseteq P$ by testing for all constraints $c' \in P$ if the length of both paths $x \rightsquigarrow u \rightarrow v \rightsquigarrow y$, $-y \rightsquigarrow u \rightarrow v \rightsquigarrow -x$ are longer than d' and the sum of the upper bounds $\rho'(x) + \rho'(-y)$ is greater than d' where $(x, y, d') \in E(c')$. If all three cases hold then c' is not implied by $\phi \cup \{c\}$ thanks to Theorem 4.

The run-time is determined by the run-time of INCCONDIFF, the calculation of δ_u^+ , δ_v^+ which are $\mathcal{O}(n \log n + m)$, and the implication check $\mathcal{O}(p)$. All the other computations can be done in constant or linear time with respect to n and m . So the overall run-time is $\mathcal{O}(n \log n + m + p)$. \square

The cost of INCCONDIFF and the shortest path computations are each $\mathcal{O}(n \log n + m)$, while the implication checking is $\mathcal{O}(p)$. The space required simply stores the graph and implication constraints.

6 Experimental Results

We present empirical comparisons of the algorithms discussed herein, first on satisfaction and then on implication questions.

For both experiments we generate 60 UTVPI instances ϕ in each problem class with the following specifications: the values d range uniformly in from -15 to 100 . approximately 10% are negative, each variable appears in at least one UTVPI constraint, each constraint involves exactly two variables, and there is at most one constraint between any two variables two variables are allowed.

In addition, for the implication benchmarks 10 implication sets P of size p were created for each n using the same restrictions as defined above. On average over all benchmarks, 65% of the constraint P were implied by the corresponding ϕ .

The experiments were run on a Sun Fire T2000 running SunOS 5.10 and a 1 GHz processor. The code was written in C and compiled with gcc 3.2.

We run incremental satisfiability on a system of m constraints in n variables, adding the constraints one at a time. We compare: INCLAMU the incrementalization of LAMU presented in Section 4, ScSt the incremental implication

Table 2. Average run-time in seconds of the satisfiability algorithms

examples		INCLAMU	SCST	m LAMU	HAST
$n = 100$	feasible	0.32	0.85	1.06	2.17
$m = 1000$	Z-inf.	0.21	0.59	0.59	1.98
$d = 5\%$	Q-inf.	0.10	0.25	0.27	1.12
all (32, 8, 20)		0.23	0.62	0.74	1.79
$n = 100$	feasible	1.10	2.48	3.96	2.76
$m = 2000$	Z-inf.	0.41	1.06	1.30	2.36
$d = 10\%$	Q-inf.	0.08	0.20	0.22	0.94
all (31, 9, 20)		0.66	1.50	2.32	2.09
$n = 100$	feasible	4.02	7.35	12.62	3.22
$m = 4000$	Z-inf.	0.40	1.06	1.21	2.54
$d = 20\%$	Q-inf.	0.09	0.24	0.26	1.10
all (28, 12, 20)		1.98	3.72	6.2	2.37
$n = 200$	feasible	4.42	10.59	17.47	22.94
$m = 4000$	Z-inf.	1.08	3.15	3.30	18.29
$d = 5\%$	Q-inf.	0.34	0.88	0.95	8.70
all (30, 11, 19)		2.51	6.15	9.64	17.42
$n = 200$	feasible	16.22	31.10	55.75	26.30
$m = 8000$	Z-inf.	1.36	3.94	4.20	20.42
$d = 10\%$	Q-inf.	0.28	0.72	0.82	6.60
all (29, 11, 20)		8.18	16.00	27.99	18.66
$n = 200$	feasible	61.69	98.71	196.82	28.52
$m = 16000$	Z-inf.	1.86	5.20	5.82	24.94
$d = 20\%$	Q-inf.	0.31	0.79	0.88	7.10
all (28, 12, 20)		29.26	47.37	93.03	20.67

checking algorithm of Section 5 where $p = 0$, m LAMU running LAMU m times for m satisfaction checks, and HAST the algorithm of [6]. The results are shown in Table 2, where d represent the density of a UTVPI instance. We split the examples into cases that are feasible, \mathbb{Z} infeasible, and \mathbb{Q} infeasible. Moreover, the table entry “all” shows the overall average run-time and the number of examples for each case in the same ordering as above written. Interestingly for dense satisfiable systems HAST is best, but overall INCLAMU is the clear winner.

The incremental implication checked satisfiability and the implications of constraints P incrementally as each of the m constraints were added one at a time. A run was terminated if there were no more constraints to add, all constraints in P were implied, or unsatisfiability was detected. We compare the two algorithms that can check implication: SCST versus HAST. Table 3 shows the results. Overall the checks for implication are cheap compared to the satisfiability check for each algorithm. Hence the results are similar to the satisfiability case. Again HAST is superior for dense systems, while SCST is the clear winner on sparse systems.

Table 3. Average run-time in seconds of the implication algorithms

examples		ScST	HAST
$n = 100$	$p = 50$	0.62	1.81
$m = 1000$	$p = 100$	0.63	1.82
$d = 5\%$	$p = 200$	0.65	1.84
$n = 200$	$p = 100$	6.18	17.52
$m = 4000$	$p = 200$	6.24	17.56
$d = 5\%$	$p = 400$	6.36	17.66
$n = 800$	$p = 400$	14.20*	521.6*
$m = 12800$	$p = 800$	14.67*	522.2*
$d = 1\%$	$p = 1600$	15.60*	523.3*

* Average run-time of \mathbb{Q} infeasible problems.

7 Conclusion

We have presented new incremental algorithms for UTVPI constraint satisfaction and implication checking which improve upon the previous asymptotic complexity, and perform better in practice for sparse constraint systems.

We can easily adapt the algorithms herein to provide non-incremental implication checking in $\mathcal{O}(n^2 \log n + nm + p)$ time and $\mathcal{O}(n + m + p)$ space, and generate all implied constraints in $\mathcal{O}(n^2 \log n + nm)$ time and $\mathcal{O}(n + m + p)$ space, where p is the number of implied constraints generated.

References

1. R. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.
2. B. V. Cherkassky and A. V. Goldberg. Negative-cycle detection algorithms. In *Proceedings of the European Symposium on Algorithms*, pages 349–363, 2006.
3. S. Cotton and O. Maler. Fast and Flexible Difference Constraint Propagation for DPLL(T). In *Theory and Applications of Satisfiability Testing - SAT 2006*, volume 4121, pages 170–183. Springer-Verlag, 2006.
4. L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.
5. D. Frigioni, A. Marchetti-Spaccamela, and U. Nanni. Fully dynamic shortest paths and negative cycle detection on digraphs with arbitrary edge weights. In *European Symposium on Algorithms*, pages 320–331, 1998.
6. W. Harvey and P. J. Stuckey. A Unit Two Variable Per Inequality Integer Constraint Solver for Constraint Logic Programming. In *The 20th Australasian Computer Science Conference (Australian Computer Science Communications)*, pages 102–111, Sydney, Australia, 1997.
7. J. Jaffar, M. J. Maher, P. J. Stuckey, and R. H. C. Yap. Beyond finite domains. In *PPCP '94: Proceedings of the Second International Workshop on Principles and Practice of Constraint Programming*, pages 86–94. Springer-Verlag, 1994.

8. S. K. Lahiri and M. Musuvathi. An Efficient Decision Procedure for UTVPI Constraints. In *Frontiers of Combining Systems*, volume 3717, pages 168–183. Springer-Verlag, 2005.
9. A. Miné. The octagon abstract domain. *Higher-Order and Symbolic Computation*, 2006.
10. R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Abstract DPLL and abstract DPLL modulo theories. In *LPAR'04*, volume 3452 of *LNAI*, pages 36–50, 2004.
11. I. Sitzmann and P. Stuckey. O-trees: a constraint based index structure. In M. Orłowska, editor, *Proceedings of the Eleventh Australasian Database Conference (ADC2000)*, pages 127–135. IEEE Press, January 2000.