Constructing Cycles in the Simplex Method for DPLL(T)*

Bertram Felgenhauer[1] and Aart Middeldorp[1]

Department of Computer Science, University of Innsbruck, Innsbruck, Austria
{bertram.felgenhauer,aart.middeldorp}@uibk.ac.at

Abstract. Modern SMT solvers use a special DPLL(T) variant of the simplex algorithm to solve satisfiability problems in linear real arithmetic. Termination is guaranteed by Bland's pivot selection rule, but it is not immediately obvious that such a rule is required. For the traditional simplex method non-termination is well-understood, but the cycling examples from the literature do not immediately carry over to the DPLL(T) variant. We present two SMT encodings of the problem of finding cycles, using linear and nonlinear real arithmetic.

1 Introduction

The simplex algorithm (Dantzig 1947) is the most popular method for solving linear programs, despite its worst-case exponential complexity. Termination of the simplex algorithm is guaranteed by pivot selection strategies, like Bland's rule [6].

Dutertre and de Moura [9] proposed an adaptation of the simplex method to decide quantifier-free linear arithmetic (QF_LRA) that works well in a DPLL(T) setting and which is used in SMT solvers like Yices [8] and Z3 [5]. The correctness of the decision procedure follows from termination of the algorithm [10, Theorem 1], which relies on Bland's pivot selection rule. The algorithm is covered in the textbook [14] by Kroening and Strichman, where it is called the *general simplex algorithm*. We prefer the name *DPLL(T) simplex algorithm*, because the algorithm does not, in fact, generalize the simplex method. Teaching a course on decision procedures using this book (as well as [7]) led to the question whether Bland's pivot selection rule is essential for termination. This paper reports on our quest to answer this question.

The literature on the simplex method contains several cycling examples, e.g. [2,4,13,20], but these typically do no carry over to the DPLL(T) setting without further ado because they start from a feasible solution and the cycling behavior is triggered by the objective function, which is absent in the DPLL(T) simplex method.

We describe two new approaches to automatically find cycling examples. The first approach targets the DPLL(T) simplex method. A sequence of pivoting steps is fixed such that the induced tableau cycles. This is followed by the

* This research was supported by Austrian Science Fund (FWF) project P27528.

use of an SMT solver for linear real arithmetic to find bounds on the variables and an initial assignment for the variables such that the pivoting steps are valid. This approach works well and is able to find small cycles which are useful for didactic purposes. In the second approach the complete search is encoded into nonlinear real arithmetic, both for the DPLL(T) simplex method and the standard simplex method with Dantzig's pivoting rule. The resulting SMT problems in quantifier-free nonlinear real arithmetic are nontrivial and could serve as interesting benchmarks for SMT solvers. The code we produced while preparing this paper is available online.[1]

The remainder of the paper is organized as follows. In the next section we briefly recall the DPLL(T) simplex algorithm. In Section 3 we give two examples showing that the algorithm may cycle if Bland's pivot selection rule is violated. The first one is found by our program. The second one originates from [2] and relies on the fact that the constant vector in the linear program is zero and hence the objective function of the dual linear program is constant, which ensures that it cannot affect the cyclic behavior. In Section 4 we explain how to construct cycles using an SMT solver for linear real arithmetic to find bounds on the variables and the initial assignment, after the initial tableau and the sequence of pivoting steps is fixed. Encoding the whole search for cycles as a satisfiability problem requires solving nonlinear real arithmetic constraints. This is described in Section 5 for the DPLL(T) simplex method as well as the original simplex method. In Section 6 we comment on related work. In particular, we analyzed the examples from the survey paper by Avis *et al.* [1], where we observed violations of Dantzig's pivot selection rule and a few typographical errors. We conclude in Section 7.

2 DPLL(T) Simplex Algorithm

The DPLL(T) simplex algorithm is a constraint solving method for linear arithmetic over real (or rational) numbers $x_1, \ldots, x_n$. The unknowns $x_1, \ldots, x_n$ are divided into *basic* variables B and *nonbasic* variables N that are related as follows:

$$x_i = \sum_{j \in N} a_{ij} x_j \tag{1}$$

for all $i \in B$. Here $\{1, \ldots, n\} = B \uplus N$ and $a_{ij} \in \mathbb{R}$ for all $i \in B$ and $j \in N$. The coefficients a_{ij} form a $|B| \times |N|$ matrix which is called the *tableau* of the problem. In addition, every variable x_i with $1 \leqslant i \leqslant n$ is equipped with upper and lower bounds u_i and l_i:

$$-\infty \leqslant l_i \leqslant x_i \leqslant u_i \leqslant +\infty \tag{2}$$

The infinities signal the absence of a corresponding bound. Throughout the algorithm, an assignment for the variables is maintained such that (1) is satisfied and (2) holds for every $i \in N$. If (2) holds also for every $i \in B$ then the algorithm

[1] http://cl-informatik.uibk.ac.at/research/simplex/

returns the current, satisfying, assignment. Otherwise $i \in B$ is selected such that (2) is violated. Next a suitable $j \in N$ is selected such that x_i and x_j can be swapped in a pivoting operation. If there is no suitable $j \in N$ then the algorithm terminates and reports that the constraint problem is unsatisfiable. The index $j \in N$ is *suitable* if one of the following mutually exclusive conditions is satisfied:

(L_+) $x_i < l_i$, $a_{ij} > 0$, and $x_j < u_j$,
(L_-) $x_i < l_i$, $a_{ij} < 0$, and $l_j < x_j$,
(U_+) $u_i < x_i$, $a_{ij} > 0$, and $l_j < x_j$,
(U_-) $u_i < x_i$, $a_{ij} < 0$, and $x_j < u_j$.

Once $j \in N$ is selected, a pivoting step is performed: $B' = B \cup \{j\} - \{i\}$, $N' = N \cup \{i\} - \{j\}$, and the coefficients in (1) are updated such that

$$x_{i'} = \sum_{j' \in N'} a_{i'j'} x_{j'} \tag{3}$$

holds for all $i' \in B'$. This amounts to substituting

$$x_j = \frac{1}{a_{ij}} \left(x_i - \sum_{j' \in N - \{j\}} a_{ij'} x_{j'} \right)$$

into the previous tableau. Next the value of x_j is changed to l_i in cases (L_+) or (L_-) and to u_i in cases (U_+) or (U_-). This is followed by a recomputation of the values of the new basic variables such that (3) remains true.

Example 1. Consider the tableau

$$\begin{array}{c} \\ x_4 \\ x_5 \end{array} \begin{array}{c} \begin{array}{ccc} x_1 & x_2 & x_3 \end{array} \\ \begin{pmatrix} 2 & 2 & -1 \\ -1 & 1 & 3 \end{pmatrix} \end{array}$$

with bounds

$$x_1 \geqslant 0 \qquad x_2 \geqslant 0 \qquad x_3 \geqslant 0 \qquad x_4 \geqslant 3 \qquad x_5 \leqslant -2$$

and assignment $x_1 = x_2 = x_3 = x_4 = x_5 = 0$. Both x_4 and x_5 violate their respective bounds. We can pivot x_4 with x_1 or x_2, but not with x_3; in order to increase the value of x_4 we have to decrease the value of x_3 (due to the negative coefficient -1) but x_3 is at its lower bound. So the nonbasic variable x_3 is not suitable for the basic variable x_4 because $x_4 < l_4$, $a_{43} < 0$, but condition (L_-) above is violated. Likewise, x_5 can be pivoted with x_1, but not with x_2 or x_3. So there are three different options for a pivoting step: (x_4, x_1), (x_4, x_2), and (x_5, x_2).

In the simplex method, the selection of the pair (x_i, x_j) with $i \in B$ and $j \in N$ is determined by a *pivot selection rule* and critical for ensuring termination of the method. Different pivoting rules have been proposed in the literature (e.g. [6,11,12,16]). Termination of the DPLL(T) simplex algorithm has been established in [10] for *Bland's rule* [6], which selects the smallest $i \in B$ such that x_i violates its bounds and smallest suitable $j \in N$.

Example 2. In the preceding example, the pair (x_4, x_1) is selected by Bland's rule, resulting in the new tableau

$$\begin{array}{c} \\ x_1 \\ x_5 \end{array} \begin{array}{c} \begin{array}{ccc} x_4 & x_2 & x_3 \end{array} \\ \begin{pmatrix} \frac{1}{2} & -1 & \frac{1}{2} \\ -\frac{1}{2} & 2 & \frac{5}{2} \end{pmatrix} \end{array}$$

and the assignment $x_1 = \frac{3}{2}$, $x_2 = x_3 = 0$, $x_4 = 3$, $x_5 = -\frac{3}{2}$.

3 Two Cycles

We give two examples where the DPLL(T) simplex algorithm may loop if one does not impose any constraints on pivots beyond suitability. The first one is obtained by the method that we describe in the next section.

Example 3. We use four variables, x_1 to x_4, with the following constraints:

$$\begin{array}{lll} x_3 = x_1 + 2x_2 & -1 \leqslant x_1 \leqslant 0 & -5 \leqslant x_3 \leqslant -4 \\ x_4 = 2x_1 + x_2 & -4 \leqslant x_2 \leqslant 0 & -7 \leqslant x_4 \leqslant 1 \end{array}$$

The resulting cycle is given in Figure 1, with the pivoting element indicated in each tableau. Note that after the first four steps (which are given in the left column), the tableaux repeat, but the assignments are different. In fact, any nonbasic variable that is at its lower bound will be at its upper bound fours steps later, and vice versa. Figure 2 displays the trajectory of the (x_1, x_2) coordinates of the eight assignments, along with the lines corresponding to the lower and upper bounds of each variable. Every assignment lies at the intersection of two of these lines, because in each step of the cycle, the two nonbasic variables are at one of their bounds. Each pair of subsequent assignments lie on one of those lines, determined by the nonbasic variable that is *not* pivoted in the corresponding pivoting step. It is noteworthy that the trajectory alternates between left and right turns, a behavior already observed by Beale [4]. The second step violates Bland's pivot selection rule as the basic variable x_1, which precedes the selected basic variable x_4, also violates its bounds. The nonbasic variable x_2 is suitable for pivoting with x_1, and the resulting pivoting step produces the tableau

$$\begin{array}{c} \\ x_3 \\ x_1 \end{array} \begin{array}{c} \begin{array}{cc} x_2 & x_4 \end{array} \\ \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{3}{2} & -\frac{1}{2} \end{pmatrix} \end{array}$$

and assignment $x_1 = -1$, $x_2 = -\frac{3}{2}$, $x_3 = -4$, and $x_4 = -\frac{7}{2}$, which satisfies the constraints. A simpler satisfying assignment is $x_1 = -1$, $x_2 = -2$, $x_3 = -5$, and $x_4 = -4$.

Note that the selection of nonbasic variables follows Bland's rule when using the (natural) variable ordering $x_1 < x_2 < x_3 < x_4$. If one instead considers the variable ordering $x_4 < x_1 < x_2 < x_3$, then the selection of basic variables follows

Bland's rule. However, the third pivoting step, which pivots the basic variable x_1 with the nonbasic variable x_3, should pivot x_1 with x_4 instead, because x_4 precedes x_3 in this variable ordering. Consequently, both parts of Bland's pivot selection rule are required in order to ensure termination.

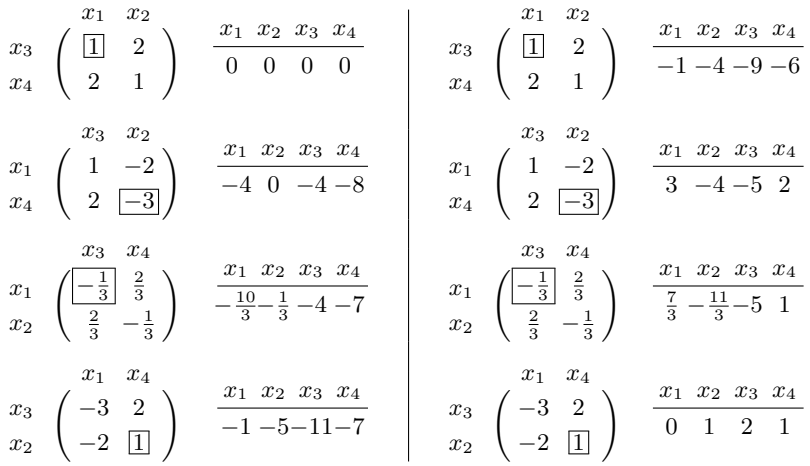

Fig. 1. The cycle of length 8 in Example 3.

The second example is an adaptation of an example attributed to H.W. Kuhn in [2].

Example 4. The linear program in [2, Example 2] (without the third row) amounts to:

$$x_4 = -2x_1 + \tfrac{1}{3}x_2 - 2 \qquad x_6 = x_1 - \tfrac{1}{3}x_2 + 1 \qquad \max u = 0x_1 + 0x_2$$
$$x_5 = -9x_1 + x_2 - 3 \qquad x_7 = 9x_1 - x_2 + 12 \qquad x_i \geqslant 0$$

Using the transformation $x_3 := x_4 + 2$, $x_4 := x_5 + 3$, $x_5 := x_6 - 1$, $x_6 := x_7 - 12$ we obtain the following constraints:

$$x_3 = -2x_1 + \tfrac{1}{3}x_2 \qquad x_5 = x_1 - \tfrac{1}{3}x_2 \qquad x_1 \geqslant 0 \qquad x_3 \geqslant 2 \qquad x_5 \geqslant -1$$
$$x_4 = -9x_1 + x_2 \qquad x_6 = 9x_1 - x_2 \qquad x_2 \geqslant 0 \qquad x_4 \geqslant 3 \qquad x_6 \geqslant -12$$

With these constraints, a cycle of length 6 is obtained. However, some of the beauty of the original example is lost, where after just two pivoting steps, the tableau becomes identical to the initial tableau under a cyclic shift of the variables.

In order to restore this symmetry, we have to assign lower bounds such that all odd numbered variables have the same lower bound, and all even numbered

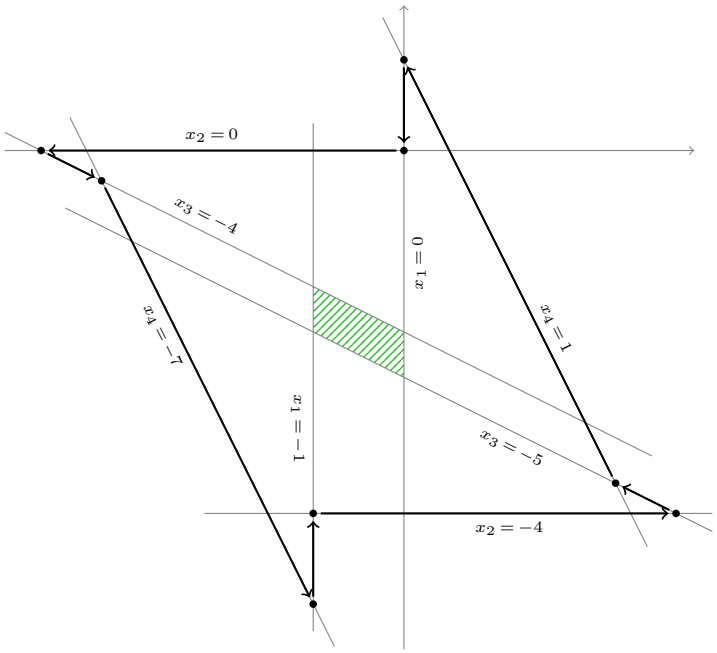

Fig. 2. Trajectory of the assignments (x_1, x_2) in Example 3.

variables as well. A suitable choice is

$$x_3 = -2x_1 + \tfrac{1}{3}x_2 \qquad x_5 = x_1 - \tfrac{1}{3}x_2 \qquad x_1, x_3, x_5 \geqslant \tfrac{1}{3}$$
$$x_4 = -9x_1 + x_2 \qquad x_6 = 9x_1 - x_2 \qquad x_2, x_4, x_6 \geqslant -3$$

The first two steps of the corresponding cycle of length 6 are given in Figure 3, after which a cyclic shift of the variables is obtained.

4 Constructing Cycles using Linear Real Arithmetic

Our first approach to finding cycles in the DPLL(T) simplex method works by fixing the initial tableau and a sequence of pivoting steps, and then setting up an SMT problem to find bounds on the variables and an initial assignment such that the pivoting steps cycle. If the resulting SMT problem is satisfiable then we have found a cycle; otherwise we try again with a different initial tableau. The procedure is outlined in Figure 4.

Our goal is to obtain a cycle with simple pivoting steps, that is, we want the tableau entries corresponding to the pivoting steps to be simple.[2] In order to control these entries we take a detour via *full tableaux*. The full tableau

[2] We use $\max(|p|, |q|)$ as a measure of simplicity of a rational number p/q; the smaller the measure the simpler the number.

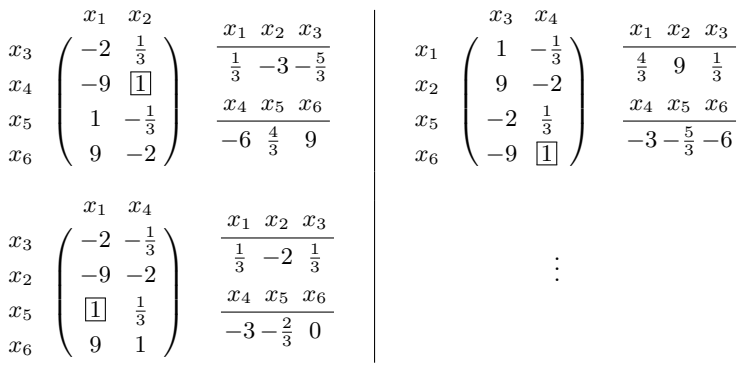

Fig. 3. One third of a cycle of length 6 in Example 4.

1: fix sequence of pivoting steps
2: **repeat**
3: guess initial tableau
4: generate linear real arithmetic SMT problem for bounds and initial assignment
5: **until** SMT problem is satisfiable with a cycle of desired simplicity
6: print cycle based on model generated by SMT solver

Fig. 4. Constructing cycles using linear real arithmetic

corresponding to

$$x_i = \sum_{j \in N} a_{ij} x_j$$

for $i \in B$ is the $|B| \times n$ matrix with entries a_{ij} where a_{ij} for $j \in B$ is defined as -1 if $i = j$ and 0 otherwise. That is, the full tableau is obtained from the (shortened) tableau by adjoining a negated identity matrix in the columns corresponding to the basic variables.

Full tableaux allow an elegant description of pivoting steps: From the tableau A, where the basic variable x_i being pivoted corresponds to a column equal to a negated unit vector $-e_k$ (we use e_k to denote the unit vector of length $|N|$ whose k-th entry equals 1), we use row operations to move to the tableau RA, where R differs from the identity matrix in the i-th column. The matrix R is chosen such that the column corresponding to the variable entering the basis in RA equals $-e_k$.

Example 5. The full tableau corresponding to the initial tableau in Example 3 is

$$\begin{array}{c} \\ x_3 \\ x_4 \end{array} \begin{array}{c} \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \end{array} \\ \begin{pmatrix} 1 & 2 & -1 & 0 \\ 2 & 1 & 0 & -1 \end{pmatrix} \end{array}$$

The first pivoting step corresponds to subtracting 2 times the first row from the second row and multiplying the first row by -1 in the full tableau, i.e.,

$$\begin{pmatrix} -1 & 0 \\ -2 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & -1 & 0 \\ 2 & 1 & 0 & -1 \end{pmatrix} = \begin{pmatrix} -1 & -2 & 1 & 0 \\ 0 & -3 & 2 & -1 \end{pmatrix}$$

Now if we assume that there is a cycle with initial full tableau A and pivoting steps given by R_1 to R_n, then from

$$R_n R_{n-1} \cdots R_1 A = A$$

we may conclude that $R_n R_{n-1} \cdots R_1 = I$ by focusing on the variables that are initially basic. Conversely, given $R_n R_{n-1} \cdots R_1 = I$ with R_i of the shape described above, we can construct a corresponding initial tableau under the additional assumption that the variable entering the basis in the j-th pivoting step is x_j. In that case, the j-th column $a_{\bullet j}$ of A satisfies

$$R_j R_{j-1} \cdots R_1 a_{\bullet j} = -e_k$$

where k is the index of the column of R_j that differs from the identity matrix. Consequently,

$$a_{\bullet j} = R_n R_{n-1} \cdots R_1 a_{\bullet j} = -R_n R_{n-1} \cdots R_{j+1} e_k$$

This gives us the desired control over the pivoting elements, whose inverse can be found on the diagonals of the R_i in the non-unit columns.

Example 6. The following four matrices satisfy $R_4 R_3 R_2 R_1 = I$:

$$R_1 = \begin{pmatrix} -1 & 0 \\ -2 & 1 \end{pmatrix} \qquad R_2 = \begin{pmatrix} 1 & -\frac{2}{3} \\ 0 & \frac{1}{3} \end{pmatrix} \qquad R_3 = \begin{pmatrix} 3 & 0 \\ 2 & 1 \end{pmatrix} \qquad R_4 = \begin{pmatrix} 1 & -2 \\ 0 & -1 \end{pmatrix}$$

We have

$$-R_4 R_3 R_2 e_1 = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \quad -R_4 R_3 e_2 = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad -R_4 e_1 = \begin{pmatrix} -1 \\ 0 \end{pmatrix} \quad -e_2 = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$$

and the resulting full tableau

$$\begin{pmatrix} 1 & 2 & -1 & 0 \\ 2 & 1 & 0 & -1 \end{pmatrix}$$

is the one underlying Example 3.

For the SMT encoding, we generate variables for the lower and upper bounds l_i and u_i and the assignments x_{ri} after each round r (using $r = 0$ for the initial assignment). For each round, we assert that the assignment satisfies the initial tableau, that the nonbasic variables that remain nonbasic do not change their value, that the pivoting step is suitable, and that the new nonbasic variable is assigned its lower bound in cases (L_+) and (L_-) of the suitability conditions

or its upper bound in cases (U_+) and (U_-). Finally we assert that the final assignment equals the initial one.

We automated the above process for the case of two constraints, with pivoting steps alternating between the first and the second row. All but two of the matrices R_i are generated randomly, picking coefficients from the set of simple numbers $\{p/q \mid |p|, |q| < 4\}$. The final two matrices are computed from the constraint $R_n R_{n-1} \dots R_1 = I$: Denoting the unknown coefficients of R_2 and R_1 by a, b, c, and d, the constraint can be written as

$$(R_n R_{n-1} \dots R_3)^{-1} = R_2 R_1 = \begin{pmatrix} 1 & a \\ 0 & b \end{pmatrix} \begin{pmatrix} c & 0 \\ d & 1 \end{pmatrix} = \begin{pmatrix} c + ad & a \\ bd & b \end{pmatrix}$$

From this we can read off a, b, d, and c, in that order, provided that $b \neq 0$.

As a final refinement, we let the number of rounds be $2n$ instead of n. This means that we perform the sequence of pivoting steps twice. This is useful because the assignment of the nonbasic variables may alternate between lower and upper bounds. Indeed exactly this happens in Example 3. The resulting set of constraints is passed to an SMT solver (in our case, Yices [8]).

We rely on the SMT solver to produce simple numbers for the assignments, which, as evidenced by Example 3, works well enough. In order to obtain this simple cycle, the termination condition in Figure 4 takes simplicity into account. The individual SMT problems are very simple; generating a problem and solving it with Yices takes well under 0.01 seconds on a Core i7-4600U (2.1 GHz) CPU, with most of the time spent in the OS rather than doing productive work. For Example 3 we filtered out unsatisfiable problems, and discarded all answers that contain numbers greater than 11; an average run succeeded after about 2000 iterations and took under 15 seconds to complete on the mentioned hardware.

5 Constructing Cycles using Nonlinear Real Arithmetic

The approach outlined in Section 4 is slightly unsatisfactory, because it involves guessing. We also tried using support for nonlinear real arithmetic (NRA) to delegate this task to the SMT solver. We tried this for both the DPLL(T) simplex method and the standard simplex method using Dantzig's pivot selection rule.

5.1 DPLL(T) Simplex

For the DPLL(T) simplex method, we fix a sequence of pivoting steps. Here we include the information which of the bounds the basic variable being pivoted violates. Then we introduce variables for the entries of the initial tableau, the lower and upper bounds, and the assignments after each round. As in Section 4, we assert that each assignment satisfies the initial tableau, and that the basic variable chosen in each step violates its selected bound. In order to ensure suitability, we encode that the nonbasic variable changes in the right direction;

formally, if x_j denotes the value before the pivoting step and x'_j the value after the pivoting step, we assert that

$$(x_j < u_j \wedge x_j < x'_j) \vee (x_j > l_j \wedge x_j > x'_j)$$

This way we avoid having to compute the entries of the intermediate tableaux. With this approach we obtain the following examples using Yices (version 2.5.2) and Z3 (version 4.5.0):[3]

Example 7. The DPLL(T) simplex method cycles with the constraints

$$\begin{array}{lll} x_3 = x_1 + 2x_2 & -\frac{5}{8} \leqslant x_1 \leqslant -\frac{9}{16} & -4 \leqslant x_2 \leqslant 1 \\ x_4 = -x_1 - \frac{1}{2}x_2 & -\frac{9}{2} \leqslant x_3 \leqslant -4 & 0 \leqslant x_4 \leqslant 4 \end{array}$$

and initial assignment $x_1 = -\frac{5}{8}$, $x_2 = -4$, $x_3 = -\frac{69}{8}$, and $x_4 = \frac{21}{8}$. This example was obtained using Yices. Z3 produces the following cycling problem:

$$\begin{array}{lll} x_3 = x_1 + 8x_2 & \frac{1}{2} \leqslant x_1 \leqslant 2 & \frac{1}{8} \leqslant x_2 \leqslant 2 \\ x_4 = -x_1 - x_2 & 6 \leqslant x_3 \leqslant 13 & -\frac{17}{4} \leqslant x_4 \leqslant \frac{1}{8} \end{array}$$

with initial assignment $x_1 = \frac{1}{2}$, $x_2 = \frac{1}{8}$, $x_3 = \frac{3}{2}$, and $x_4 = -\frac{5}{8}$.

The numbers obtained this way are not nearly as nice as those obtained by the approach based on linear real arithmetic. We tried tweaking the result by fixing some of the bounds or tableau entries, but found that it's hard to steer Yices and Z3 towards nice solutions.

5.2 Standard Simplex

This section is concerned with the standard simplex method, which is used for solving linear optimization problems of the shape

$$\begin{array}{lll} \text{minimize} & \sum_{j \in N} c_j x_j & \\ \text{subject to} & x_i + \sum_{j \in N} a_{ij} x_j = b_i & \text{for } i \in B \\ & x_i \geqslant 0 & \text{for } i \in N \cup B \end{array}$$

where we again distinguish between basic variables x_i ($i \in B$) and nonbasic variables x_j ($j \in N$). The simplex method maintains a tableau consisting of the coefficients a_{ij}, b_i, and c_j for $i \in B$ and $j \in N$ satisfying the condition $b_i \geqslant 0$ for $i \in B$, which ensures that the assignment $x_j = 0$ for $j \in N$ and $x_i = b_i$ for $i \in B$ satisfies the constraints. If no c_j ($j \in N$) is negative then the optimum has been reached; otherwise, we select a nonbasic variable x_j corresponding to a negative

[3] While CVC4 [3] (snapshot version 2017-06-14) also has support for NRA, it cannot produce of models, making it unfit for our purposes.

c_j, and look for a basic variable x_i such that a_{ij} is positive. (If there is no such basic variable then the problem is unbounded.) These two constraints determine the *suitable* pivoting pairs. To pivot x_i and x_j, we use the substitution

$$x_j = \frac{1}{a_{ij}} \left(b_i - x_i - \sum_{j' \in N - \{j\}} a_{ij'} x_{j'} \right)$$

in the given tableau and the objective function. There are several pivot selection rules for the standard simplex method. One is Bland's rule, which ensures termination; it picks the smallest (nonbasic) variable index among the negative values c_j, and the smallest index among the basic variables that are suitable for pivoting with the selected nonbasic variable.

Here we focus on Dantzig's traditional pivoting rule, which tries to reduce the value of the objective function as quickly as possible. This is the oldest pivoting rule, and it is of interest to us because it fails to ensure termination. To this end, the nonbasic variable x_j is selected such that the value c_j is as small as possible (ties are broken by variable index). The basic variable x_i is selected such that $\frac{b_i}{a_{ij}}$ is minimized. Again, ties are broken in favor of smaller variable indices.

For the standard simplex method, we first tried a very naive encoding in an attempt to find cycles of length 6. However, the trick from Section 5.1, to express all constraints in terms of the initial tableau, does not work here, because pivot selection depends on the objective function (which has to be expressed in terms of the nonbasic variables) and the pivoting elements in the tableau. Therefore, as a first attempt, we introduced variables for all tableau entries for each rounds in the assumed cycle, which were defined in terms of the pivoting element and the preceding tableau. As a simplification, which is inspired by Avis *et al.* [1], we assume that $b_i = 0$ for $i \in B$, so the minimization of $\frac{b_i}{a_{ij}}$ has no effect.

This turned out to be too much for Yices and Z3, perhaps because of the large number of functions and variables (for 6 pivoting steps and 6 variables in the tableau, the encoding requires 98 real variables for coefficients and tableau entries) and we did not obtain any cycling tableau from this encoding.

In the end we adopted an approach by Zörnig [20], which avoids divisions and requires fewer variables (26 instead of 98 for a cycle of length 6 with 6 variables). In this approach, all constraints are expressed in terms of the initial tableau using subdeterminants. We demonstrate this by an example, where we try to find a cycling tableau with two constraints and six variables $x_1, \ldots, x_6$ where the basic variables follow the cycle

$$x_5, x_6 \to x_1, x_6 \to x_1, x_2 \to x_3, x_2 \to x_3, x_4 \to x_5, x_4 \to x_5, x_6 \to \cdots$$

We can express the initial tableau by

$$A\boldsymbol{x} = \left(\begin{array}{cccccc} a_{11} & a_{12} & a_{13} & a_{14} & 1 & 0 \\ a_{21} & a_{22} & a_{23} & a_{24} & 0 & 1 \\ \hline c_1 & c_2 & c_3 & c_4 & 0 & 0 \end{array}\right) \boldsymbol{x} = \left(\begin{array}{c} 0 \\ 0 \\ \hline z \end{array}\right)$$

where $\boldsymbol{x} = (x_1, x_2, x_3, x_4, x_5, x_6)^T$, z is the value of the objective function, and we have already set $b_i = 0$ for $i \in \{1, 2\}$. In order to obtain the tableau with basic variables x_i, x_j, we have to perform row operations that result in the corresponding columns to become unit vectors; this can be expressed as

$$A'\boldsymbol{x} = \begin{pmatrix} a_{1i} & a_{1j} & 0 \\ a_{2i} & a_{2j} & 0 \\ c_i & c_j & 1 \end{pmatrix}^{-1} \left(\begin{array}{cccccc} a_{11} & a_{12} & a_{13} & a_{14} & 1 & 0 \\ a_{21} & a_{22} & a_{23} & a_{24} & 0 & 1 \\ \hline c_1 & c_2 & c_3 & c_4 & 0 & 0 \end{array}\right) \boldsymbol{x} = \left(\begin{array}{c} 0 \\ 0 \\ \hline z \end{array}\right)$$

Following Zörnig, let D_{ij} and $\overline{D}_{ijk}$ be defined as

$$D_{ij} = \begin{vmatrix} a_{1i} & a_{1j} \\ a_{2i} & a_{2j} \end{vmatrix} \qquad\qquad \overline{D}_{ijk} = \begin{vmatrix} a_{1i} & a_{1j} & a_{1k} \\ a_{2i} & a_{2j} & a_{2k} \\ c_i & c_j & c_k \end{vmatrix}$$

Using what is essentially Cramer's rule, the k-th column of the tableau A' equals

$$\begin{pmatrix} a'_{1k} \\ a'_{2k} \\ c'_k \end{pmatrix} = \frac{1}{D_{ij}} \begin{pmatrix} D_{kj} \\ D_{ik} \\ \overline{D}_{ijk} \end{pmatrix}$$

Now, for the pivoting step from x_i, x_j to x_k, x_j to be valid, the following constraints need to be satisfied:

- $a'_{1,k} > 0$: D_{ij} and D_{kj} have equal signs, and $D_{ij} \neq 0$
- $c'_k < 0$: $\overline{D}_{ijk}$ and D_{ij} have opposite signs
- c'_k is minimal, and ties are broken by smaller variable index: if $D_{ij} > 0$ ($D_{ij} < 0$), then $\overline{D}_{ijk'} > \overline{D}_{ijk}$ ($\overline{D}_{ijk'} < \overline{D}_{ijk}$) for $k' < k$ and $\overline{D}_{ijk'} \geqslant \overline{D}_{ijk}$ ($\overline{D}_{ijk'} \leqslant \overline{D}_{ijk}$) for $k' > k$.
- no variable with smaller index can enter the basis: if $j < k$ then $a'_{2,k} < 0$, i.e., D_{ij} and D_{ik} have opposite signs.

Analogous conditions can be derived for pivoting steps from x_i, x_j to x_i, x_k. In fact by the first constraint, we will always have $D_{ij} > 0$, because $D_{56} = 1 > 0$ in the initial tableau. Using this encoding the following example is obtained using Yices; Z3 did not produce an answer within 600 seconds.

Example 8. The following optimization problem cycles when using the standard simplex method with Dantzig's pivot selection rule:

$$\begin{array}{ll} \text{minimize} & -6x_1 + 46x_2 + 7x_3 + \frac{97}{32}x_4 \\ \text{subject to} & -x_5 = \frac{1}{2}x_1 - 4x_2 - \frac{3}{4}x_3 - \frac{25}{64}x_4 \\ & -x_6 = 4x_2 + x_3 + \frac{1}{2}x_4 \\ & x_1, \ldots, x_6 \geqslant 0 \end{array}$$

With some manual tweaking (by fixing some of the values of the tableau in the SMT encoding), we obtain the following, nicer optimization problem:

$$\begin{aligned} \text{minimize} \quad & -\tfrac{3}{4}x_1 + 4x_2 + x_3 + \tfrac{5}{13}x_4 \\ \text{subject to} \quad & -x_5 = \tfrac{1}{3}x_1 - 2x_2 - \tfrac{2}{3}x_3 - \tfrac{1}{3}x_4 \\ & -x_6 = 2x_2 + x_3 + \tfrac{6}{13}x_4 \\ & x_1, \ldots, x_6 \geqslant 0 \end{aligned}$$

The cycle is given in Figure 5.

$$\begin{array}{c|c|c|c} & x_1 \; x_2 \; x_3 \; x_4 & x_5 \; x_6 \; x_1 \; x_4 & x_5 \; x_2 \; x_3 \; x_4 \\ -x_5 & \left(\begin{array}{cccc} \boxed{\frac{1}{3}} & -2 & -\frac{2}{3} & -\frac{1}{3} \\ 0 & 2 & 1 & \frac{6}{13} \end{array}\right) & \left(\begin{array}{cccc} 3 & 3 & 1 & \frac{5}{13} \\ -\frac{3}{2} & -1 & -\frac{1}{2} & \boxed{\frac{1}{26}} \end{array}\right) & \left(\begin{array}{cccc} 3 & -6 & -2 & -1 \\ 0 & \boxed{2} & 1 & \frac{6}{13} \end{array}\right) \\ -x_6 & & & \\ \hline \text{min:} & \left(\begin{array}{cccc} -\frac{3}{4} & 4 & 1 & \frac{5}{13} \end{array}\right) & \left(\begin{array}{cccc} 3 & 1 & \frac{1}{4} & -\frac{2}{13} \end{array}\right) & \left(\begin{array}{cccc} \frac{9}{4} & -\frac{1}{2} & -\frac{1}{2} & -\frac{19}{52} \end{array}\right) \\ & x_5 \; x_6 \; x_3 \; x_4 & x_5 \; x_6 \; x_1 \; x_2 & x_3 \; x_6 \; x_1 \; x_2 \\ -x_1 & \left(\begin{array}{cccc} 3 & 3 & \boxed{1} & \frac{5}{13} \\ 0 & \frac{1}{2} & \frac{1}{2} & \frac{3}{13} \end{array}\right) & \left(\begin{array}{cccc} \boxed{18} & 13 & 6 & -10 \\ -39 & -26 & -13 & 26 \end{array}\right) & \left(\begin{array}{cccc} \frac{1}{18} & \frac{13}{18} & \frac{1}{3} & -\frac{5}{9} \\ \frac{13}{6} & \boxed{\frac{13}{6}} & 0 & \frac{13}{3} \end{array}\right) \\ -x_2 & & & \\ \hline & \left(\begin{array}{cccc} \frac{9}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \end{array}\right) & \left(\begin{array}{cccc} -3 & -3 & -\frac{7}{4} & 4 \end{array}\right) & \left(\begin{array}{cccc} \frac{1}{6} & -\frac{5}{6} & -\frac{3}{4} & \frac{7}{3} \end{array}\right) \end{array}$$

Fig. 5. The cycle of length 6 in Example 8.

In contrast to the encoding in Section 4, the SMT encodings for this section were produced manually. This is made feasible by the fact that the SMT-LIB format supports function definitions. For example, we defined an abbreviation for the determinant of a 3×3 matrix:

```
(define-fun det3
 ((c11 Real) (c12 Real) (c13 Real)
  (c21 Real) (c22 Real) (c23 Real)
  (c31 Real) (c32 Real) (c33 Real))
 Real
 (- (+ (* c11 c22 c33) (* c12 c23 c31) (* c13 c21 c32))
    (+ (* c11 c23 c32) (* c12 c21 c33) (* c13 c22 c31))))
```

6 Related Work

Perhaps the closest related work is Zörnig's paper [20], which explores the idea of employing a computer program for finding cycles in the standard simplex

method. Zörnig uses LINGO (a commercial program for nonlinear optimization) for this purpose. His work goes into a different direction from ours. Whereas we are mainly interested in the DPLL(T) simplex method, Zörnig explores several pivot selection rules and also synthesizes a cycle of odd length. The latter requires working with more than 2 constraints, cf. [20, Equation (5.6)].

There is a rich body of literature about loops in the standard simplex method, starting with Hoffmann [13] and Beale [4]. A survey of cycles with visualizations of the trajectories was produced by Avis *et al.* [1]. As far as we know, all of these examples have been found manually. As part of this work, we reconstructed the cycles presented in the latter paper. Below we make a few observations.

- The example attributed to Beale does not appear in [4], where instead the following cycling tableau is given:

$$\begin{aligned} \text{minimize} \quad & -\tfrac{3}{4}x_1 + 20x_2 - \tfrac{1}{2}x_3 + 6x_4 \\ \text{subject to} \quad & \tfrac{1}{4}x_1 - 8x_2 - x_3 + 9x_4 + x_5 = 0 \\ & \tfrac{1}{2}x_1 - 12x_2 - \tfrac{1}{2}x_3 + 3x_4 + x_6 = 0 \end{aligned}$$

- There are sign errors in the objective functions of Beale's example ($+150x_2$ should be $-150x_2$) and Marshall and Suurballe's example [15] (which should read $z = 0 + x_3 - 7x_4 - 1x_5 - 2x_6$).
- The examples by Sierksma [17], Yudin and Gol'shtein [19], and Solow [18] violate Dantzig's pivot selection rule; in the last case, only the tie-breaking rule for the basic variable is violated. Solow's cycle is similar to Kuhn's in that it is based on two pivoting steps that result in a cyclic shift of the variables. After four steps, the first tableau of Figure 6 is reached. Then x_1 is pivoted with x_5. In the sixth step, x_2 is pivoted with x_6, but by Dantzig's rule we should select x_1 as the new basic variable instead of x_6.

$$\begin{array}{c|cccc} & x_1 & x_2 & x_3 & x_4 \\ -x_5 & \boxed{2} & 1 & -3 & -1 \\ -x_6 & -7 & -3 & 7 & 2 \\ \hline \text{min:} & -2 & -2 & 8 & 2 \end{array} \qquad \begin{array}{c|cccc} & x_5 & x_6 & x_3 & x_4 \\ -x_1 & -3 & -1 & \boxed{2} & 1 \\ -x_2 & 7 & 2 & -7 & -3 \\ \hline & 8 & 2 & -2 & -2 \end{array}$$

$$\begin{array}{c|cccc} & x_5 & x_2 & x_3 & x_4 \\ -x_1 & \frac{1}{2} & \frac{1}{2} & -\frac{3}{2} & -\frac{1}{2} \\ -x_6 & \frac{7}{2} & \boxed{\frac{1}{2}} & -\frac{7}{2} & -\frac{3}{2} \\ \hline & 1 & -1 & 5 & 1 \end{array} \qquad \vdots$$

Fig. 6. One third of Solow's cycle of length 6 [18]

7 Conclusion

In this paper we have presented a new approach for finding cycles in the simplex method, both for the traditional method and, for the first time, for the DPLL(T) variant which is used in SMT solvers to solve quantifier-free linear arithmetic constraint problems.

Any cycle in the simplex method induces a cycle in the dual simplex method by switching to the dual optimization problem. The absence of an objective function means that this observation does not immediately carry over to the DPLL(T) simplex method. However, if a cycling tableau has shape $x_B + Ax_N = 0$, which is the case for all examples collected by Avis *et al.* [1], then dualization produces a constant objective function, and in this case, the cycle can be reproduced in the DPLL(T) simplex method. We have seen this in Example 4. It is noteworthy that this approach cannot produce Example 3, which relies on the fact that every variable comes with two constraints. In fact, Beale [4] observed that any cycle in the standard simplex method with two constraints requires at least 6 steps, and correspondingly, 6 variables, because each bound $x_i \geqslant 0$ produces one of the potential lines the trajectory can move along. Our 4 variable example works because for each variable, we get a pair of two parallel lines that can be used on the trajectory. Beale's first observation remains valid; indeed our cycle length 8 is greater than 6.

While working on Zörnig's encoding for the standard simplex we noticed that sometimes small changes (like switching from a strict inequality to a nonstrict one) resulted in Yices taking a longer time than we were willing to wait; we did not study this phenomenon systematically, but this suggests that these encodings are interesting benchmarks for nonlinear real arithmetic.

Acknowledgements. We thank the reviewers for their constructive feedback.

References

1. Avis, D., Kaluzny, B., Titley-Péloquin, D.: Visualizing and constructing cycles in the simplex method. Operations Research 56(2), 512–518 (2008), doi: 10.1287/opre.1070.0474
2. Balinski, M.L., Tucker, A.W.: Duality theory of linear programs: A constructive approach with applications. SIAM Review 11(3), 347–377 (1969), doi: 10.1137/1011060
3. Barrett, C., Conway, C.L., Deters, M., Hadarean, L., Jovanović, D., King, T., Reynolds, A., Tinelli, C.: CVC4. In: Gopalakrishnan, G., Qadeer, S. (eds.) Proc. 23rd International Conference on Computer Aided Verification. Lecture Notes in Computer Science, vol. 6806, pp. 171–177 (2011), doi: 10.1007/978-3-642-22110-1_14
4. Beale, E.M.L.: Cycling in the dual simplex algorithm. Naval Research Logistics Quaterly 2, 269–275 (1955), doi: 10.1002/nav.3800020406
5. Björner, N., de Moura, L.: Z3: An efficient SMT solver. In: Ramakrishnan, C., Rehof, J. (eds.) Proc. 14th International Conference on Tools and Algorithms for

the Construction and Analysis of Systems. Lecture Notes in Computer Science, vol. 4963, pp. 337–340 (2008), doi: 10.1007/978-3-540-78800-3_24
6. Bland, R.G.: New finite pivoting rules for the simplex method. Mathematics of Operation Research 2(2), 103–107 (1977), doi: 10.1287/moor.2.2.103
7. Bradley, A.R., Manna, Z.: The Calculus of Computation – Decision Procedures with Applications to Verification. Springer (2007), doi: 10.1007/978-3-540-74113-8
8. Dutertre, B.: Yices 2.2. In: Biere, A., Bloem, R. (eds.) Proc. 26th International Conference on Computer Aided Verification. Lecture Notes in Computer Science, vol. 8559, pp. 737–744 (2014), doi: 10.1007/978-3-319-08867-9_49
9. Dutertre, B., de Moura, L.: A fast linear-arithmetic solver for DPLL(T). In: Ball, T., Jones, R.B. (eds.) Proc. 18th International Conference on Computer Aided Verification. Lecture Notes in Computer Science, vol. 4144, pp. 81–94 (2006), doi: 10.1007/11817963_11
10. Dutertre, B., de Moura, L.: Integrating simplex with DPLL(T). Tech. Rep. SRI-CSL-06-01, SRI International (2006)
11. Goldfarb, D., Reid, J.K.: A practicable steepest-edge simplex algorithm. Mathematical Programming 12(1), 361–371 (1977), doi: 10.1007/BF01593804
12. Harris, P.M.J.: Pivot selection methods of the Devex LP code. Mathematical Programming 5(1), 1–28 (1973), doi: 10.1007/BF01580108
13. Hoffman, A.J.: Cycling in the simplex algorithm. Tech. Rep. 2974, National Bureau of Standards (1953)
14. Kroening, D., Strichman, O.: Decision Procedures – An Algorithmic Point of View. Springer (2008), doi: 10.1007/978-3-540-74105-3
15. Marshall, K., Suurballe, J.: A note on cycling in the simplex method. Naval Research Logistics Quaterly 16(1), 121–137 (1969), doi: 10.1002/nav.3800160110
16. Pan, P.Q.: A largest-distance pivot rule for the simplex algorithm. European Journal of Operational Research 187(2), 393–402 (2008), doi: 10.1016/j.ejor.2007.03.026
17. Sierksma, G.: Linear and Integer Programming. Marcel Dekker Inc., 2nd edn. (1996)
18. Solow, D.: Linear Programming: An Introduction to Finite Improvement Algorithms. North-Holland (1984)
19. Yudin, D.B., Gol'shtein, E.G.: Linear Programming. Israel Program of Scientific Translations (1965)
20. Zörnig, P.: Systematic construction of examples for cycling in the simplex method. Computers & Operations Research 33(8), 2247–2262 (2006), doi: 10.1016/j.cor.2005.02.001