Detecting Cycles in Dynamic Graphs in Polynomial Time (Preliminary Version)

S. Rao Kosaraju*

Gregory F. Sullivan

Dept. of Computer Science, Johns Hopkins Univ., Baltimore, MD 21218

Abstract

Consider a digraph which has labels on its edges which are k-dimensional vectors. In this paper we show it is possible in polynomial time to determine if such a digraph contains a zero cycle, i.e., a cycle whose edge labels sum to the zero vector component-wise. This solves the open problem of finding cycles in dynamic graphs which was posed by Iwano and Steiglitz. Our solution has a time complexity of $O(|V|\log(|V|)Z)$ where Z is the complexity of a linear programming problem. For the important cases of two and three dimensions we present $O(Z)$ time algorithms. The linear programming problems we solve have at most $2|E|$ variables and $O(|E|+|V|+k)$ constraints.

1 Introduction

Many VLSI applications involving regular structure can be modeled by 2- and 3- dimensional dynamic graphs [2,1]. A k-dimensional dynamic digraph is an infinite digraph obtained by copying a basic digraph in a k-dimensional orthogonal grid. The nodes in each cell of the grid are connected to a finite number of other cells according to a fixed pattern. This infinite periodic digraph can be represented by a digraph with k-dimensional vector labels on its edges [4,2]. Digraphs with vector labels are sometimes called "static" digraphs to contrast them with the "dynamic" digraphs they are derived from. A dynamic digraph has a cycle iff its corresponding static digraph has a zero cycle. It was established by Iwano and Steiglitz [2] that zero cycle detection in 2-dimensional static digraphs is of considerable practical significance. They developed polynomial time algorithms for various special cases of 2-dimensional static digraphs and left open the general case. Here we solve the general problem by developing a polynomial time algorithm for any static digraph with k-dimensional labels. Further, we develop techniques for 2 and 3 dimensions to significantly reduce our time complexity. We first develop an $O(|V||E|Z)$ step algorithm for the general k-dimensional problem, where Z is the complexity of linear programming problems of a type we shall describe. We then use some additional ideas to modify and refine this algorithm to derive an $O(|V|\log(|V|)Z)$ step algorithm. Next, we investigate the nature of zero cycles in the practically significant cases of 2 and 3 dimensions. Several ideas enable us to solve the two dimensional case with an overall expense of 3 linear programming problems and an all-pairs shortest path computation. These ideas also enable us to solve the three dimensional case with an expense of $O(1)$ linear programming problems and shortest path computations. In section 2 we state the terminology and the linear programming framework. In sec-

*Research supported by the National Science Foundation under Grant DCR-8506361.

tion 3 we give a polynomial time algorithm for zero cycle detection in k-dimensional digraphs. In sections 4 and 5 we give tighter bounds for zero cycle detection in two and three dimensional digraphs respectively. In section 6 we conclude with some open problems.

2 Linear Programming Framework

Definition 2.1 Let $(G(V,E),T)$ be a digraph with k-dimensional labels on each edge, i.e., let T be a function from E to $\mathbf{Q}^k$ where $\mathbf{Q}^k$ is the set of all k-vectors of rational numbers. A *zero cycle* of (G,T) is a cycle $\langle e_1,\ldots,e_q\rangle$ in G such that $\sum_{i=1}^{q} T(e_i) = 0_k$, where 0_k is the zero vector of k elements and the summation is a componentwise vector sum. (The cycle need not be simple). We extend the definition of T to apply to sequences of edges such that $T(C) = \sum_{i=1}^{q} T(e_i)$ when $C = \langle e_1,\ldots,e_q\rangle$. We extend T further to apply to sets of cycles $\mathbf{C}$ such that $T(\mathbf{C}) = \sum_{C\in\mathbf{C}} T(C)$. A *zero multicycle* of a labeled digraph is defined to be a non-null set of cycles $\mathbf{C}$ such that $T(\mathbf{C}) = 0_k$.

We initially solve a relaxed version of the zero cycle problem.

Theorem 2.2 *Determining if a k-dimensional labeled digraph has a zero multicycle can be done in polynomial time.*

Proof: Let (G,T) be a labeled digraph. Define a variable for each edge of the digraph, x_e for $e \in E$. The following four equations define the problem. (1) For $v \in V$, $\sum_{e\in IN(v)} x_e = \sum_{e\in OUT(v)} x_e$. (2) For $e \in E$, $x_e \geq 0$. (3) $\sum_{e\in E} x_e T(e) = 0_k$. (4) $\sum_{e\in E} x_e \geq 1$. (IN(v) and OUT(v) refer to the in- and out-edges of v, respectively).

Suppose (G,T) has a zero multicycle. Let s_e be the number of times edge e occurs in the multicycle. We claim that letting $x_e = s_e$ is a solution for the linear programming problem . Equation (1) is satisfied since for each vertex v, if a cycle contains an occurrence of an edge in $IN(v)$ then that edge can be paired with an occurrence of an edge in $OUT(v)$ from the same cycle. Equation (2) is satisfied immediately by our solution construction. Equation (3) is satisfied because our construction is based on a zero multicycle. Equation (4) is satisfied because the multicycle contains at least one edge.

Suppose equations (1)-(4) are satisfiable and let s_e, $e \in E$ be a solution. We claim (G,T) has a zero multicycle. First, we note that any positive multiple of a solution is also a solution. The existence of any solution implies the existence of a rational solution and by scaling we may further assume an integral solution. By equation (2) these integers are non-negative and by equation (4) at least one is positive. Consider the multigraph defined by making s_e copies of each edge e. Equation (1) allows us to claim the multigraph is balanced, i.e., $IN(v) = OUT(v)$ for $v \in V$. It follows that each weakly connected component has an eulerian cycle. These cycles correspond to a multicycle in the original digraph. By equation (3) it is a zero multicycle. ∎

Definition 2.3 The first three equations of the linear programming problem defined above are called the *standard constraints*. Let a complete formulation, as in the two theorems immediately below, be called a *multicycle LP*.

Theorem 2.4 *Let (G,T) be a labeled digraph and let LP be a linear programming problem with standard constraints and constraint $\sum_{e\in E'} x_e \geq 1$ for some subset of edges E'. LP has a solution iff (G,T) has a zero multicycle which contains an element of E'.*

Theorem 2.5 *Let (G,T) be a labeled digraph. Let $\mathbf{E}$ be a set of subsets of edges. Let LP be a linear programming problem with the standard constraints and the following set of constraints: For each $E' \in \mathbf{E}$, $\sum_{e\in E'} x_e \geq 1$. If LP has a* *solution then there exists a zero multicycle which contains at least one edge from each $E' \in \mathbf{E}$. If LP has no solution then there exists an E' in $\mathbf{E}$ such that no edge in E' is contained in any zero multicycle.*

Definition 2.6 If $(G(V,E),T)$ is a labeled digraph and $P \subseteq V$ then (G_P,T_P) is the labeled digraph *induced* by P which is defined as follows: the vertex set of G_P is P and the edge set consists of all edges which are incident only with vertices in P. T_P is T with its domain restricted to the edges of G_P.

Definition 2.7 Let $G(V,E)$ be a digraph with $E' \subseteq E$. We say E' induces subgraph $G'(V',E')$ where V' contains the vertices which are incident with some edge of E'.

Definition 2.8 Let $O(Z)$ be the time complexity of a multicycle LP for (G,T). Note, the LP has $|E|$ variables and the standard constraints use $|V|+|E|+k$ equations. The LP problems defined in the two theorems above use either one additional equation or $|\mathbf{E}|$ additional equations. In our algorithms $|\mathbf{E}|$ is bounded by $|V|$. Vaidya [5] has shown that linear programming has a time complexity of $O(((m+n)n^2+(m+n)^{1.5}n)L)$ where m is the number of equations, n is the number of variables and L is the number of bits in the representation of the problem. See also [3].

3 Zero Cycles in k-Dimensional Labeled Digraphs

Definition 3.1 We define a relation on the vertices of a labeled digraph, (G,T). $v_1 \sim v_2$ iff $v_1 = v_2$ or v_1 and v_2 are in the same cycle of some zero multicycle for (G,T). It is not difficult to show that $\sim$ is an equivalence relation. Let $(G,T)^\sim$ be the partition of vertices defined by $\sim$.

Definition 3.2 We say a partition is trivial if it contains one element. Otherwise, we say it is non-trivial. Let V be a set and let $\mathbf{P}$ and $\mathbf{P}'$ be partitions of V. We say $\mathbf{P}$ is *finer* than $\mathbf{P}'$ iff for all $P' \in \mathbf{P}'$ there exists $P \in \mathbf{P}$ such that $P \subseteq P'$. If $\mathbf{P}$ is finer than $\mathbf{P}'$ then we say $\mathbf{P}'$ is coarser than $\mathbf{P}$. Note that the partition into single element sets is finer than all other partitions. Also, a partition is finer than itself and coarser than itself.

Our first technique for solving zero cycle detection in k-dimensional digraphs depends on computing $(G,T)^\sim$ and using it to recursively decompose the problem. The partition can be computed in polynomial time by a simple method which uses $|E|$ linear programs. Let us define a subset of edges: $E' = \{e : e$ is contained in some zero multicycle for $(G,T)\}$. E' can be computed by using $|E|$ linear programs.

For each edge e we attempt to solve the linear program with the standard constraints and the constraint $x_e \geq 1$. We put e in E' iff the linear program has a solution. Let $G'(V',E')$ be the digraph induced by E'. One can show that $v_1 \sim v_2$ iff there is a path from v_1 to v_2 in G'. Thus, the partition defined by $\sim$ is identical with the vertex partition defined by the strongly connected components of G'. If we call this method of computing $(G,T)^\sim$ VPARTITION then the following ZCYCLE algorithm solves the zero cycle problem.

Algorithm ZCYCLE(G,T)

Input: k-dimensional labeled digraph (G,T).
Output: Yes, if (G,T) contains a zero cycle;
No, otherwise.

```
1. IF $|V| = 1$ THEN IF $(G,T)$ has a zero multicycle
     THEN RETURN Yes ELSE RETURN No
2. CALL VPARTITION($G,T$) and LET $\mathbf{P}$ be the
     vertex partition returned
3. IF $|\mathbf{P}| = 1$ THEN RETURN Yes
4. FOR EACH $P \in \mathbf{P}$ DO
```

```
5.  CALL ZCYCLE($G_P, T_P$)
6.  IF ZCYCLE returns Yes THEN
        RETURN Yes
7. END FOR EACH
8. RETURN No
```

Theorem 3.3 *Algorithm ZCYCLE correctly determines if a digraph contains a zero cycle in $O(Z|V||E|)$ time where $O(Z)$ is the time complexity of a multicycle LP.*

Proof: Correctness can be proved with an induction on the number of vertices in V. The depth of recursion of ZCYCLE is bounded by $|V|$ since the number of vertices of a digraph in a recursive call decreases by at least one. Consider the set of recursive calls made at depth i. The arguments of these calls are all disjoint subgraphs of (G,T). Thus, the time complexity of all the calls at level i is bounded by the complexity of a single call with argument (G,T) (not counting work at the next level). There are $|V|$ levels and the work on each level is dominated by the calls to VPARTITION which have a total complexity of $O(Z|E|)$. The overall complexity is $O(Z|V||E|)$ ▮

In the following we show that the task of VPARTITION can be reimagined and its algorithm redesigned so that it requires $O(\log(|V|))$ linear programs instead of $O(|V|)$. It does not compute $(G,T)^\sim$ exactly. Instead, it computes a partition satisfying the following two weaker constraints: One, it is coarser than $(G,T)^\sim$. Two, it is non-trivial if $(G,T)^\sim$ is non-trivial. The modified VPARTITION is given below.

Definition 3.4 If $\mathbf{P}$ is a partition of vertices and w is a vertex then $\mathbf{P}(w)$ is the element of the partition which contains w. (Additional commands after BREAK in the body of the loop are skipped and the next iteration begins immediately).

Algorithm VPARTITION(G,T)

```
$\mathbf{P} := \{\{v_1\},\{v_2\},\ldots,\{v_{|V|}\}\}$
REPEAT DO
    FOR EACH $P \in \mathbf{P}$ DO
        $OUT_P := \{e : \text{tail}(e) \in P$ and
            $\text{head}(e) \in P'$ and $P' \neq P$ and $P' \in \mathbf{P}\}$
    END FOR EACH
    IF FOR EACH $P \in \mathbf{P}$, $OUT_P = \emptyset$ THEN
        RETURN $\mathbf{P}$
    DEFINE a linear programming problem, LP,
        with standard constraints and the following:
        For each $P \in \mathbf{P}$, $\sum_{e \in OUT_P} x_e \geq 1$
    IF LP has no solution THEN DO
        LOCATE NON-MERGING ELEMENTS $\mathbf{Q}$
        RETURN $\mathbf{Q} \cup \{v : v \in V$ and there does not
            exist $Q \in \mathbf{Q}$ with $v \in Q\}$.
    END IF
    IF LP has a solution THEN DO
        APPROPRIATE MERGES
END REPEAT
```

The algorithm starts with a partition $\mathbf{P}$ which is the finest possible. It performs a sequence of merges on the elements of $\mathbf{P}$ while maintaining the invariant that $\mathbf{P}$ is finer than $(G,T)^\sim$. This repeats until $\mathbf{P} = (G,T)^\sim$ or $\mathbf{P}$ contains one or more elements which appear in $(G,T)^\sim$. If $\mathbf{P} = (G,T)^\sim$ then the algorithm simply returns $\mathbf{P}$. Otherwise, the algorithm must locate some of the elements which appear in $(G,T)^\sim$. These elements are called non-merging elements. When a non-null subset of them, $\mathbf{Q}$, is found the algorithm returns $\mathbf{Q} \cup \{v : v \in V$ and there does not exist $Q \in \mathbf{Q}$ with $v \in Q\}$. This satisfies the output specification.

The merging of elements in $\mathbf{P}$ occurs at DO APPROPRIATE MERGES based on the solution to the linear programming problem LP. If s_e, $e \in E$ is a solution to LP then one can show that for every $s_e > 0$ the partition elements containing the tail and head of edge e can be safely merged. The merges occurring at each iteration of the algorithm based on an LP solution will at least halve the number of elements in $\mathbf{P}$. At most $\log(|V|)$ rounds of merging occur before either $\mathbf{P} = (G,T)^{\sim}$ or non-merging elements are detected. If LP has no solution then one can show there exist *non-merging elements* $\mathbf{Q} \subseteq \mathbf{P}$.

To find these elements a new more complex strategy is employed. It is also based on maintaining a partition. However, the partition is not based on vertices; it is based on the partition elements of $\mathbf{P}$. Initially, the partition is defined as $\mathcal{Q} = \{\{P\} : P \in \mathbf{P}\}$. We perform a sequence of merges and deletions of the elements of $\mathcal{Q}$ until it contains a single element. This element will contain some of the non-merging elements of the partition $\mathbf{P}$ and only non-merging elements. During a single iteration of this process the elements of $\mathcal{Q}$ are paired together and merged to form $\mathcal{D}\mathcal{Q}$. A linear program is executed and the results allow us to either delete at least half the elements of $\mathcal{Q}$ or to reassign $\mathcal{Q}$ to be $\mathcal{D}\mathcal{Q}$. We maintain the invariant that at least one of the elements of $\mathcal{Q}$ contains only non-merging elements of $\mathbf{P}$. After $O(\log(|V|)$ iterations we have found some of the non-merging elements of $\mathbf{P}$. Thus we can establish the following theorem.

Theorem 3.5 *The modified algorithm VPARTITION has a time complexity of $O(\log(|V|)Z)$ and can be used with ZCYCLE to detect zero cycles with time complexity $O(Z|V|\log(|V|))$ where $O(Z)$ is the time complexity of a multicycle LP.*

4 Zero Cycles in Two Dimensional Labeled Digraphs

Definition 4.1 We define a relation on the vertices of a digraph with one-dimensional labels, (G,T).
$v_1 \bowtie v_2$ iff $v_1 = v_2$ or v_1 and v_2 are together in some zero cycle of (G,T), i.e., there exists a cycle C with $T(C) = 0$ and v_1 and v_2 are both in C. Note, $\bowtie$ is an equivalence relation.

Definition 4.2 Let (G,T) be a digraph with one dimensional labels and let C be a cycle. We say C is a negative, positive or zero cycle if we have $T(C) < 0$, $T(C) > 0$ or $T(C) = 0$, respectively.

It is possible to design an algorithm which we call V1PARTITION(G,T) which determines the partition above for a restricted case. It takes as input a 1-dimensional labeled digraph (G,T) which either contains no negative cycles or no positive cycles. It then outputs the vertex partition defined by $\bowtie$. Algorithm V1PARTITION can be implemented with a time complexity of $O(|V|^3)$. It is based on an easy modification of the all pairs shortest path algorithm.

Definition 4.3 Let $A = (x_1, y_1)$ and $B = (x_2, y_2)$ be vectors in $\mathbb{Q}^2$. We define the projection of A onto B to be $project(A,B) = (A \cdot B)/\|B\|$. The dot product is defined by $(x_1,y_1)\cdot(x_2,y_2) = x_1x_2 + y_1y_2$. The norm is defined by $\|(x_1,y_1)\| = \sqrt{x_1^2 + y_1^2}$. When calculating *project* in our algorithms we can avoid the square root calculation by letting $project(A,B) = A \cdot B$. This is acceptable because we only test sums of vectors which have been projected onto a common vector. We only test whether these sums are positive, negative or zero.

The following algorithm detects the existence of zero cycles. If the given digraph is not strongly connected, the algorithm will be applied to each strongly connected component.

Algorithm Z2CYCLE(G,T)

```
Input: 2-dimensional labeled digraph $(G,T)$
  which is strongly connected.
Output: Yes, if $(G,T)$ contains a zero cycle;
  No, otherwise.

1 IF $(G,T)$ does not contain a zero multicycle
    THEN RETURN No
2 LET $C$ be a cycle of some zero multicycle
3 IF $T(C) = 0_2$ THEN RETURN Yes
4 LET b be some non-zero vector perpendicular
```

```
   to $T(C)$
5 FOR ALL $e \in E$, $T'(e) := project(T(e), b)$
6 IF $(G, T')$ has a positive cycle and a negative
   cycle THEN RETURN Yes
7 CALL V1PARTITION$(G, T')$ and LET $\mathbf{P}$ be
   the returned partition
8 FOR EACH $P \in \mathbf{P}$ DO
9   IF $(G_P, T_P)$ has a zero multicycle then
      RETURN Yes
10END FOR EACH
11RETURN No
```

Theorem 4.4 *Algorithm Z2CYCLE is correct and has a time complexity of $O(Z)$ where $O(Z)$ is the time complexity of a multicycle LP.*

Proof: For the first step it is clear that if G has no zero multicycle then G has no zero cycle. For the second and third steps we note that if $T(C) = 0_2$ then we have found a zero cycle. Let b and T' be as defined in the algorithm at lines four and five. Now suppose (G, T') has negative and positive cycles. Call them C_{neg} and C_{pos}. We will show that (G, T) has a zero cycle. Let D be a cycle which includes every vertex of G. This cycle exists since G is strongly connected. If $project(T(D), b)$ is negative then one can combine multiple copies of D and C_{pos} to yield a single cycle D' with $project(T(D'), b) = 0$. Similarly, if $project(T(D), b)$ is positive then one can combine copies of D and C_{neg}. Hence, there is a cycle D' with $project(T(D'), b) = 0$. Since C is part of a multicycle we know there is a collection of cycles $\mathbf{C}$ such that $T(\mathbf{C}) = -T(C)$. Since b is perpendicular to $T(C)$ we can combine copies of C, $\mathbf{C}$ and D' to yield a single zero cycle. Therefore if step six of the algorithm returns yes then (G, T) does contain a zero cycle.

Now suppose execution reaches step seven. We know (G, T') contains only non-negative cycles or it contains only non-positive cycles. We claim the partition $\mathbf{P}$ is the same partition as the one defined by $\sim$ on (G, T). If $u \sim v$ then there is a cycle D containing u and v which is part of a zero multicycle $\mathbf{D}$. Note $project(T(D), b)$ must be zero. Suppose it were positive then (G, T') must have only non-negative cycles. Thus, it would follow that $project(T(\mathbf{D}), b)$ would be positive - a contradiction. A similar contradiction holds if the projection were negative and shows $project(T(D), b)$ must be zero. We may conclude $u \bowtie v$. Suppose $u \bowtie v$ then there is a cycle D containing u and v for which $project(T(D), b) = 0$. We can use multiple copies of D, C and $\mathbf{C}$ to yield a zero multicycle. We may conclude $u \sim v$.

When execution reaches step eight we claim (G, T) has a zero cycle iff (G_P, T_P) contains a zero cycle for some $P \in \mathbf{P}$. This follows because there cannot be a zero cycle containing vertices from different elements of $\mathbf{P}$. We next claim and show that (G_P, T_P) contains a zero cycle iff it contains a zero multicycle. One direction is immediate. For the other direction we know that any cycle D that is part of a zero multicycle must have $project(T(D), b) = 0$. Equivalently, $T(D)$ is a multiple of $T(C)$. This fact allows us to construct a single cycle, X, in a digraph (G_P, T_P) which contains every vertex of G_P and is a multiple of $T(C)$. Suppose (G_P, T_P) contains a zero multicycle $\mathbf{K}$. Then $\mathbf{K}$ may contain a zero cycle in which case we are done. Alternatively, $\mathbf{K}$ may contain cycles K_{pos} and K_{neg} such that $T(K_{pos})$ is a positive multiple of $T(C)$ and $T(K_{neg})$ is a negative multiple. Copies of K_{pos}, K_{neg} and X can be combined to yield a zero cycle in G_P. We conclude that testing for zero multicycles in step nine also tests for zero cycles. Finally, we conclude the algorithm is correct. ▮

We can modify ZCYCLES so that it only makes at most 3 calls to a linear programming routine by replacing lines 8,9 and 10. Rather than make one call for each component only two calls are made total. The components are segregated into two sets POS and NEG according to the rule that follows: A zero cycle D from each component (G_P, T_P') is tested. If we determine that project$(T(D), T(C)) > 0$ then the component is put in POS. If project$(T(D), T(C)) < 0$

then the component is put in NEG. (If it equals zero then a zero cycle for (G,T) has been found.) One can show (G,T) contains a zero cycle iff the digraph consisting of all the POS components has a zero cycle or the digraph consisting of all the NEG components has a zero cycle.

5 Zero Cycles in Three Dimensional Labeled Digraphs

To find cycles in 3-dimensional dynamic digraphs we introduce two techniques in this section. We define a semibasis and a cut plane and show how to find them in polynomial time for small dimensions. The semibasis we construct is for the set CSPAN(G,T) defined below.

Definition 5.1 Let (G,T) be a k-dimensional labeled digraph. CSPAN$(G,T) = \{rT(\mathrm{C})$: where C is a multicycle in G and $r \geq 0$ is a real number $\}$

Definition 5.2 Given a set $S \subseteq \mathrm{R}^k$ we say S is a *cone* iff there exists a set of vectors Y in R^k such that $S = \{x : x = \sum_{y \in Y} k_y y$, where for each $y \in Y$, k_y is a nonnegative real$\}$. We say Y *generates* the cone S.

Note, CSPAN(G,T) is a cone where $Y = \{T(C) : C$ is a simple cycle in $G\}$. If we are able to show that CSPAN(G,T) contains certain sets of vectors then we may conclude (G,T) has a zero cycle. Specifically, if G is strongly connected and there are k linearly independent vectors, b_i, $1 \leq i \leq k$, such that the pairs of vectors b_i, $-b_i$, $1 \leq i \leq k$ are in CSPAN(G,T) then we may conclude (G,T) has a zero cycle. (Note, this is equivalent to saying CSPAN$(G,T) = \mathrm{R}^k$). This motivates the finding of a semibasis defined below.

Definition 5.3 Let $S \subseteq \mathrm{R}^k$ be a cone. We say B is a ***semibasis*** of S iff for all $b \in B$, $b \in S$ and $-b \in S$; the elements of B are linearly independent; and B has maximum cardinality.

One can incrementally construct a semibasis for CSPAN(G,T). The principle tool used is described in the theorem below. Suppose one has constructed a set B' which is a subset of some semibasis B. If $|B'| < |B|$ then one can find a vector to add to B by solving some linear programs. Let AUG be a set of vectors such that $B \cup$ AUG spans R^k and each element of AUG is orthogonal to every element of B'. Execute a set of linear programs of the type defined below by letting b be each element of AUG in turn. If one of these problems has a solution then one can extract from the solution a simple cycle C such that $T(C)$ can be added to B'. If none of the linear programs has a solution one can show B' is a semibasis. For a 3-dimensional problem this method allows us to find a semibasis in $O(Z)$ steps.

Theorem 5.4 *Let (G,T) be a k-dimensional labeled digraph. Let b be a k-dimensional vector. We can answer the following question in polynomial time: Does (G,T) contain multicycles* C *and* C$'$ *such that $T(\mathrm{C}) = -T(\mathrm{C}')$ and $T(\mathrm{C})$ is not orthogonal to b?*

Proof: In the degenerate case where $|E| = 0$ the answer is no. Otherwise we can set up a linear program with two variables, x_e and y_e, for each edge e of the digraph. One can show the following equations define the problem. (1) For $v \in V$, $\sum_{e \in IN(v)} x_e = \sum_{e \in OUT(v)} x_e$ and $\sum_{e \in IN(v)} y_e = \sum_{e \in OUT(v)} y_e$. (2) For $e \in E$, $x_e \geq 0$ and $y_e \geq 0$. (3) $\sum_{e \in E}(x_e + y_e)T(e) = 0_k$. (4) $\sum_{e \in E} x_e \geq 1$. (5) $\sum_{e \in E} x_e T(e) \cdot b \geq 1$. ∎

We now define cut planes. These planes can be used to recursively decompose our cycle problem.

Definition 5.5 Let $S \subseteq \mathrm{R}^k$ and $b \in \mathrm{Q}^k$. We say b defines a *cut plane* for S iff for every $s \in S$, $b \cdot s \geq 0$ or for every $s \in S$, $b \cdot s \leq 0$. Note, the defining vector is normal to the cut plane itself.

We are interested in cut planes of the cone defined by CSPAN(G,T). The decomposition works as follows: If b defines a cut plane then setting $T'(e) = project(T(e),b)$ for each $e \in E$ gives a one dimensional labeling. Note, (G,T') can not contain both positive and negative cycles. We can use the algorithm V1PARTITION to compute the partition $\mathbf{P}$ defined by $\bowtie$. It follows that (G,T) has a zero cycle iff for some $P \in \mathbf{P}$, (G_P, T_P) has a zero cycle.

Finding cut planes is non-trivial. In a two dimensional space a cone S has a simple structure. S can be generated by a set Y with $|Y| \leq 3$. Further, if S is not the whole plane or a half plane then there is a set Y with $|Y| \leq 2$. The algorithm we give for zero cycle detection in 3 dimensions requires the calculation of a cut plane when a digraph (G,T) has a semibasis of cardinality one, $\{b\}$. Intuitively, CSPAN(G,T) in this case looks like a "trough". The intersection of CSPAN(G,T) with the plane normal to b looks like a "V". The theorem below allows us to find vectors Y which can be used to find a cut plane. Specifically, one can show that if $Y = \{y_1, y_2\}$ then $(y_1 - y_2) \times b$ defines a cut plane.

Theorem 5.6 *Let (G,T) be a 3-dimensional labeled digraph with semibasis $\{b\}$. In polynomial time we can find a set Y of linearly independent vectors with $|Y| \leq 2$ such that the set $Y \cup \{b, -b\}$ generates the set* CSPAN(G,T)

Proof: Let $\{b_1, b_2\}$ be two vectors such that the vectors $\{b, b_1, b_2\}$ are mutually orthogonal. Consider the equations: Objective function (1) $\sum_{e\in E} x_e T(e) \cdot b_j$. (2) For $v \in V$, $\sum_{e \in IN(v)} x_e = \sum_{e\in OUT(v)} x_e$. (3) For $e \in E$, $x_e \geq 0$. (4) $\sum_{e\in E} x_e \geq 1$. (5) $\sum_{e \in E} x_e T(e) \cdot b_i = q$. (6) $\sum_{e\in E} x_e T(e) \cdot b = 0$. There are eight variations of this linear program which are obtained as follows: (a) Maximizing or minimizing the objective function. (b) Letting q equal 1 or -1. (c) Letting $i = 1$, $j = 2$ or $i = 2$, $j = 1$. One can show that these variations allow us to find the Y vectors which generate the cone. ▌

We now sketch the algorithm for zero cycle detection in 3 dimensions. Our first step is to compute a semibasis B for the input (G,T). If $|B| = 3$ then clearly there is a zero cycle. If we ever find a semibasis of cardinality zero then we can easily test for zero cycles since they are present iff zero multicycles are present.

If $|B| = 1$ then we know CSPAN(G,T) is shaped like a "trough" as in the discussion above. The solution of a small number of linear programs allows us to find a vector b defining a cut plane. Our choice of b assures that the cut plane intersects with CSPAN(G,T) to form only a line. Let (G_P, T_P) be an instance of the decomposition of the problem based on b as discussed above. One can prove that (G_P, T_P) has a zero cycle iff it has a zero multicycle.

Now, consider the problem when $|B| = 2$. Let $B = \{b_1, b_2\}$. One can show CSPAN(G,T) is either a plane or a plane together with every point on one side of the plane. Let b be the vector orthogonal to b_1 and b_2 and note that b defines a cut plane. Let (G_P, T_P) be an instance of the decomposition of the problem based on b. If (G_P, T_P) still has a semibasis of size 2 then one can show it contains a zero cycle. If $|B| = 0$ then we can immediately test for zero cycles. This leaves the case in which $|B| = 1$ which we showed how to solve immediately above. Thus it is possible to establish the following theorem.

Theorem 5.7 *Cycles in three dimensional labeled digraphs can be detected in $O(Z)$ time.*

6 Conclusions and Open Problems

For the lower dimensional cases such as 2- and 3-, can we solve the problem without using linear programming. We conjecture that such solutions exist since multicycle linear programing problems are very similar to network flow and circulation formulations. For the k dimensional case, how few linear programs can one use to solve the problem. Can the complexity of the k-dimensional case be expressed as a polynomial function of k times the complexity of a linear program.

Acknowledgements: Thanks to Mike McKenna for his many helpful suggestions.

References

[1] K. Iwano and K. Steiglitz. Optimization of one-bit full adders embedded in regular structures. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1289–1300, 1986.

[2] K. Iwano and K. Steiglitz. Testing for cycles in infinite graphs with periodic structure. In *Proc. of 19th ACM STOC*, pages 46–55, 1987.

[3] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[4] J. Orlin. Some problems on dynamic / periodic graphs. In W. R. Pulleyblank, editor, *Progress in Combinatorial Optimization*, pages 273–293, Academic Press, Orlando, Florida, 1984.

[5] P. M. Vaidya. An algorithm for linear programming which requires $O(((m+n)n^2 + (m+n)^{1.5}n)L)$ arithmetic operations. In *Proc. of 19th ACM STOC*, pages 29–38, 1987.