A POLYNOMIAL-TIME ALGORITHM TO FIND THE SHORTEST CYCLE BASIS OF A GRAPH

J.D. HORTON
SCHOOL OF COMPUTER SCIENCE
UNIVERSITY OF NEW BRUNSWICK
FREDERICTON, N.B. E3B 5A3

TR84-026, AUGUST 1984

A POLYNOMIAL-TIME ALGORITHM TO FIND THE SHORTEST CYCLE BASIS OF A GRAPH

J.D. Horton
School of Computer Science
University of New Brunswick
FREDERICTON, N.B. E3B 5A3

A POLYNOMIAL-TIME ALGORITHM TO FIND THE SHORTEST CYCLE BASIS OF A GRAPH

ABSTRACT

Define the length of a basis of a cycle space to be the sum of the lengths of all circuits in the basis. An algorithm is given that finds a basis with the shortest length in $O(e^3v)$ operations. Edges may be weighted or unweighted.

A POLYNOMIAL-TIME ALGORITHM TO FIND THE SHORTEST CYCLE BASIS OF A GRAPH

1. Introduction

Given a graph (V,E) with weights on the edges, how can one find the basis of the cycle space with minimum total weight? This question may have some uses in Surveying [6], [1]. Algorithms that attempt to answer this question have been developed in [6] and [4]. Hubicka and Syslo [4] conjectured that their algorithm B works, but Kolasinska has recently constructed a counterexample [5]. Steeves in [6] developed an algorithm that takes $O(e\ v^2)$ operations, but counterexamples have also been found for it as well.

A second use for minimum cycle bases may be to improve algorithms that list all simple circuits in a graph. One early reference and one recent reference for this type of algorithm are [8] and [7]. Dixon and Goodman use a similar technique to search for the longest cycle in a graph [2].

The main result of this paper is to give an algorithm that solves this problem $O(e^3\ v)$ operations.

2. Definitions

In this paper, graphs are finite, undirected, without loops or multiple edges. Although not strictly necessary, we can also assume graphs are always connected. An edge is denoted by the (unordered) pair of its endpoints. Each edge e of a graph is weighted with a real number w(e), which extends to a weight function on all sets of edges. The

unweighted case or equivalently the case in which all weights are 1, does not seem to lead to any simplification of the algorithm. Weights are allowed to be negative, but the weight of a simple circuit (a connected subgraph which is regular of degree 2) cannot be allowed to be negative, as in the shortest path problem.

A cycle is any set of edges of a graph in which each vertex of the graph is incident with an even number of edges. Two cycles, C and D, can be added to form their sum, C+D, using the set symmetric difference operation, $(C \cup D) - (C \cap D)$. Equivalently, one can identify a cycle with its incidence vector, and cycle-addition as vector addition over the integers modulo 2. The set of all cycles, which is closed under addition, is called the cycle space, and is generated by the set of cycles corresponding to the simple circuits.

Let the number of vertices in the graph be v; let the number of edges in the graph be e. Also, let d=e-v+1 be the dimension of the cycle space.

3. Theoretical Results

A cycle in the minimum cycle basis of a graph must have certain properties, which are developed in this section. Then only circuits that have these properties need to be considered by the algorithm.

Lemma. If $\underline{B}$ is a cycle basis for a graph, $C \in \underline{B}$, and $C = C_1+C_2$, then $\underline{B}-\{C\} \cup \{C_i\}$ is a cycle basis for one of i=1 or 2.

Proof. Both C_1 and C_2 are generated by $\underline{B}$, and each in a unique way. Hence for both i=1 and 2, C_i is the sum of a subset of the cycles in $\underline{B}$, say $\underline{A}_i$, and $\underline{A}_i$ is unique. Since (the sum of cycles in $\underline{A}_1$) = $C_1 = C+C_2$ = C + (the sum of cycles in $\underline{A}_2$), $\underline{A}_1$ and $\underline{A}_2$ differ only in that one of them includes C and the other does not. Assume $C \varepsilon \underline{A}_1$. Then $\underline{B}-\{C\}$ generates C_1, hence $\underline{B}' = \underline{B} \cup \{C_2\}-\{C\}$ generates all cycles in $\underline{B}$, and is a basis.

Theorem 1. A minimum cycle basis always consists of simple circuits.

Proof. Assume $\underline{B}$ is a cycle basis, C is in $\underline{B}$, and C is not a simple circuit. Then C contains a simple circuit C_1, and $C = C_1+C_2 = C_1 \cup C_2$, where C_2 is a cycle. If all simple circuits have positive weight, $0 < w(C_1) < w(C)$, and $0 < w(C_2) = w(C)-w(C_1) < w(C)$. By the lemma, $\underline{B}' = \underline{B}-\{C\} \cup \{C_i\}$ is a basis for one of i=1 or 2. But $\underline{B}'$ has less weight than $\underline{B}$.

Define P(x,y) to be the shortest (minimum weight) path from vertex x to vertex y in G.

Theorem 2. Let x and y be two vertices in a circuit C of a minimum cycle basis $\underline{B}$. Then P(x,y) is contained in C.

Proof. Let P_1 and P_2 be the two paths joining x and y in C. Assume $P(x,y) \neq P_1$ and $P(x,y) \neq P_2$. Define $C_i = P(x,y)+P_i$ for i=1 and 2. Then C_1 and C_2 are cycles, and $C = C_1+C_2$. Note that

$w(c_i) \leq w(P(x,y))+w(P_i) < w(P_1)+w(P_2) = w(C)$. By the lemma, $\underline{B}' = \underline{B}-\{C\}$ $\{C_i\}$ for i=1 or 2 is also a basis, and hence B is not minimum.

Theorem 3. Let x be any vertex of any circuit C in a minimum cycle basis $\underline{B}$. Then there is an edge (y,z) in C such that C = P(x,y)+P(x,z)+(y,z).

Proof. Let $C = (x_0,x_1,\ldots,x_{n-1},x_n)$ where $x = x_0 = x_n$. Define $P_i = (x,x_1,\ldots,x_i)$ and $Q_i = (x,x_{n-1},\ldots x_i)$. By theorem 2, $P(x,x_i) = P_i$ or Q_i. If $P(x,x_i) = P_i$ and $1 \leq j \leq i$, then $P(x,x_j) = P_j$ because a subpath of any shortest path is also a shortest path. Similarly, if $P(x,x_i) = Q_i$ and $n>j\geq i$, then $P(x,x_j) = Q_j$. Let $y=x_i$ where i is the largest index such that $P(x,x_i) = P_i$, and let $z = x_{i+1}$. Then $P(x,y)+(y,z)+P(x,z) = P_i+(x_i,x_{i+1})+Q_{i+1} = C$.

Theorem 3 gives a strong condition on the circuits in a minimum cycle basis. All basis circuits can be found by considering for each vertex, edge pair (x,(y,z)), the circuit C(x,y,z) = P(x,y)+P(x,z)+(y,z). Obviously there are at most ve such circuits. The set of cycles form a matroid so that the greedy algorithm can be used to find the minimum cycle basis.

Another result that might be useful in a practical implementation is that the shortest circuit through an edge is always in the circuit basis. This fact is used in [4] and its predecessors.

Theorem 4. If e is an edge, then the shortest circuit C_e through e is in the minimum cycle basis $\underline{B}$.

Proof. If C_e is not in $\underline{B}$, then C_e is the sum of some cycles in $\underline{B}$. Consider any cycle C in this sum such that e is in C. Then $C = C_e + C'$, where C' is the sum of cycles in $\underline{B}-\{C\}$. Hence $\underline{B}' = \underline{B}-\{C\}\cup\{C_e\}$ is a basis by the lemma, and $\underline{B}'$ has less weight than B.

4. The Algorithm

The outline of the algorithm can be given as follows:

(1) Find minimum paths between all pairs of vertices.

(2) For each vertex v and edge (x,y) in the graph, create the circuit $C(v,x,y) = P(v,x)+P(v,y)+(x,y)$.

(3) Order the circuits by weight.

(4) Use the greedy algorithm to find the minimum cycle basis from this set of circuits.

A quick analysis indicates that step 4 is the critical step:

Step 1 takes $O(v^3)$ operations [3];

Step 2 takes $O(ev^2)$ operations;

Step 3 takes $O(n \log(n))$ operations;

Step 4 takes $O(nk)$;

where n is the number of circuits found in step 2, and k is the number of operations to decide whether a circuit is independent of another given set of circuits or not. A simple method to implement step 4 is to consider the cycles as rows of a 0-1 matrix. The columns correspond to

the edges of the graph; the rows are the incidence vectors of the circuits. Gaussian elimination using elementary row operation can be applied to the matrix. Instead of processing one column at a time, it is better to process each row in turn, in order of the weights of the circuits. Each row can be processed in O(ed) operations, where e is the number of columns, and d is the maximum number of independent circuits. Since the total number of circuits that has to be processed is O(ev), step 4 takes $O(e^2dv)$ steps.

The main point at which the algorithm can be improved is to decrease the set of circuits under consideration. The first observation is that edges in the shortest spanning tree grown from a vertex will never generate a circuit with that vertex, so the number of circuits generated is at most O(dv). The second observation is that each cycle C in the minimum cycle basis will be generated $|C|$ times, once for each vertex. The third observation is that not all circuits generated need to satisfy Theorem 3.

The circuit can be tested to see whether Theorem 3 is satisfied in O(k) operations, where k is the number of edges in the circuit to be tested. Assume the circuit is $C(x_1,x_i,x_{i+1})=(x_1,x_2,\ldots,x_k,x_1)$. For each vertex v in a circuit, define r(v) to be the vertex in the circuit furthest from v such that P(v,r(v)) follows the circuit to the right. Thus $r(x_1)=x_i$. Similarly, define l(v) to be the vertex in the circuit furthest from v such that P(v,l(v)) follows the circuit to the left. Theorem 3 states that for any circuit in the minimum cycle basis r(v) and l(v) are neighbors in the circuit for all vertices in the circuit. The functions r and l can be calculated in linear time for all vertices.

For example, for x_2 we know $P(x_2,x_i)=(x_2,x_3,\ldots x_i)$ since a subpath of a shortest path is itself a shortest path. Consider $P(x_{i+1}, X_2)$. If the second vertex of the path is x_i, then $P(x_2,x_{i+1}) = (x_2,x_3,\ldots,x_{i+1})$; else $r(x_2)=x_{i+1}$. Perform the same test on x_{i+2}, $x_{i+3},\ldots x_j$, until the test fails for x_{j+1}. Then $r(x_2)=x_j$. Now $r(x_3)$ can be calculated starting with x_{j+1}, etc. The function $1(v)$ can be calculated in the same way, with $1(x_1)=x_{i+1}$, then calculate in turn $1(x_k),1(x_{k-1}),\ldots,1(x_2)$.

It is not true that any circuit which satisfies the condition of theorem 3 is necessarily in the basis. A counterexample is given in Figure 1. The shortest cycle basis of this graph will be the set of inner faces. However, the outer face satisfies the condition of theorem 3. Thus one cannot hope that this check will eliminate the check for linear independence of the cycles. It is possible that the set of cycles will often be significantly less but I have been unable to prove it.

One problem that can arise is that the minimum weight cycles are not unique. In fact, the proof of theorem 2 assumed that the minimum length path between two vertices was unique. This can be guaranteed by perturbation or lexicographic methods. Let the minimum path be the path that contains the vertex of the lowest index. If the lowest vertices are tied, then the weight of the second lowest breaks the tie, etc. Two simple methods of handling the problem of ties are available. One is to calculate the paths whenever a tie occurs, the second is to keep track of the paths as they are created. Both methods require a factor of v to be added to the worst-case analysis of step 1. But step 4 still dominates step 1, if $d>v$.

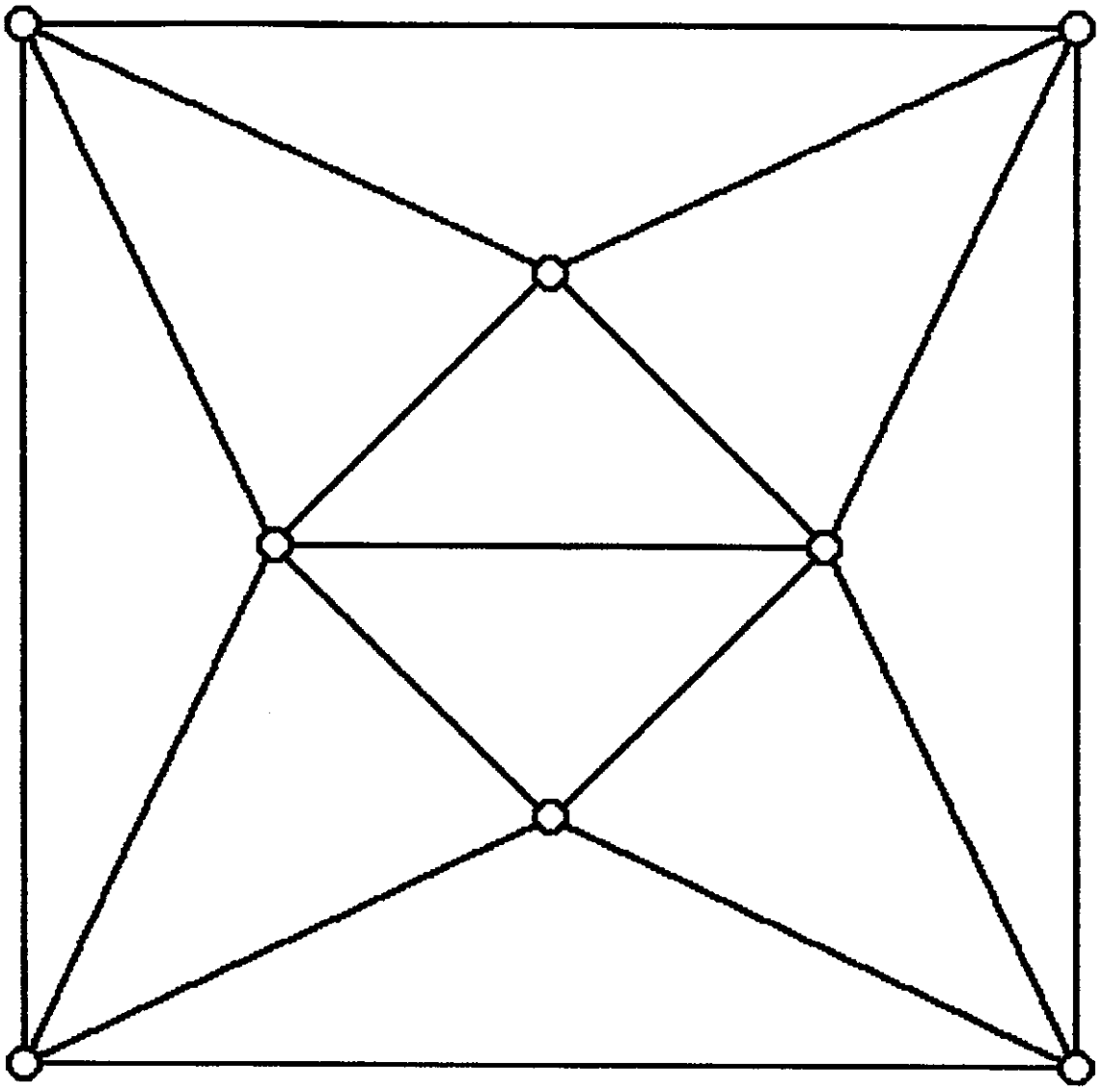

Figure 1. Counterexample

REFERENCES

1. D.W. Cribb, R.D. Ringeisen, and D.R. Shier, "On Cycle Bases of a Graph," Congressus Numerantium, 32(1981), 221-229.

2. E.T. Dixon and S.E. Goodman, "An Algorithm for the Longest Cycle Problem", Networks, 6(1976), 139-149.

3. R.W. Floyd, "Algorithm 97: Shortest Path", Comm. ACM, 5(1962), 345.

4. E. Hubicka and M.M. Syslo, "Minimal Bases of Cycles of a Graph", Recent Advances in Graph Theory, Proceedings of the Symposium held in Praque, June 1974, Academia Praha, Praque, 1975.

5. E. Kolasinska, "On a Minimum Cycle Basis of a Graph", Zastosowania Matematyki, 16(1980), 631-639.

6. P.A. Steeves, "Numerical Processing of Horizontal Control Data-Economization by Automation", unpublished Ph.D. thesis, University of New Brunswick, 1984.

7. M.M. Syslo, "An Efficient Cycle Vector Space Algorithm for Listing All Cycles of a Planar Graph", SIAM J. Comput., 10(1981), 797-808.

8. J.T. Welch Jr., "A Mechanical Analysis of the Cyclic Structure of Undirected Graphs", J. ACM, 13(1966), 205,210.

TECHNICAL REPORTS

SCHOOL OF COMPUTER SCIENCE

Number	Date	Author	Title
TR74-001	Feb 1974	L.F. Johnson	A Search Algorithm for the Simple Cycles of a Directed Graph
TR74-002	Oct 1974	W.D. Wasson R. McIssaac	A New Spanning Tree Algorithm
TR74-003	Oct 1974	U.G. Gujar	Remote Job Entry and Output Through APL
TR75-004	Apr 1975	U.G. Gujar	Subroutines with Variable Numbers of Arguments
TR75-005	Jul 1975	L.E. Garey	Block Methods for Nonlinear Volterra Integral Equations
TR75-006	Aug 1975	D.M. Fellows	Comments on "A General Fortran Emulator for IBM 360/370 Random Number Generator 'RANDU'"
TR75-007	Aug 1975	L.E. Garey M. LeBlanc	Quadrature Formulae for Functions of Two Variables and Applications
TR75-008	Sep 1975	L.F. Johnson	Determining Cliques of a Graph
TR75-009	Oct 1975	D.M. Miller	An Algorithm for Determining the Chromatic Number of a Graph
TR76-010	Jan 1976	L.E. Garey	Step by Step Methods for the Numerical Solution of Volterra Integro-Differential Equations
TR76-011	Jan 1976	U.G. Gujar	A device Independent Computer Plotting System
TR76-012	Mar 1976	P.P. Emin	A Partition Monitor for Fast-Batch-Processing with Limited Execution (Fable)
TR77-013	Dec 1976	U.G. Gujar J.A. Fitzgerald	A Driver for Raster-Like Plotting Devices
TR77-014	Jan 1977	U.G. Gujar D.M. Fellows	Automatic Job Scheduling in HASP

Number	Date	Author	Title
TR79-015	May 1979	U.G. Gujar J.M. DeDourek M.E. McIntyre	A Method for Designing a Lexical Analyzer
TR79-016	Jun 1979	U.G. Gujar A.R. Nagesh	ϕSCUBA A Buffered Core Graphics System
TR79-017	Jun 1979	T.A. Middleton	A Transformation Approach to Implementing Aggregate Operations
TR79-018	Jul 1979	T.A. Middleton	On Assignment Between Data Paths
TR79-019	Aug 1979	T.A. Middleton	On the Use of "Fixed" Data Paths for Assignment
TR79-020	Aug 1979	T.A. Middleton	Representing Data Paths by Program Structures
TR79-021	Sep 1982	J.D. Horton	Sets With No Empty Convex 7-Gons
TR83-022	Jun 1983	J.D. Horton	Resolvable Path Designs
TR83-023	Aug 1983	U.G. Gujar F.W.L. So	Computer Display of Characters
TR84-024	Jun 1984	J.D. Horton	A Lower Bound on the Number of One-Factors in Bicubic Graphs
TR84-025	Jul 1984	U.G. Gujar	3-D Graphics in APL: User Perspective
TR84-026	Aug 1984	J.D. Horton	A Polynomial-Time Algorithm to Find the Shortest Cycle Basis of a Graph