5
W49
C43
546

COMPUTER SCIENCES DEPARTMENT

University of Wisconsin - Madison

ON NON-INTERSECTING EULERIAN CIRCUITS

by

Samuel W. Bent
and
Udi Manber

Computer Sciences Technical Report #546

June 1984

On Non-Intersecting Eulerian Circuits

Samuel W. Bent
and
Udi Manber

Computer Science Department
University of Wisconsin
1210 W Dayton St.
Madison, Wisconsin 53706

June 1984

This research was supported in part by the National Science Foundation under Grants MCS-8203238 and MCS-8303134.

On Non-Intersecting Eulerian Circuits

Samuel W. Bent and Udi Manber
University of Wisconsin-Madison

ABSTRACT

The following question arises in flame-cutting and similar applications. "Given a graph drawn in the plane, is there an Eulerian circuit in which successive edges always belong to a common face?" We prove that this question and related ones are NP-complete.

1. Introduction

The problem of determining whether a graph has an Eulerian circuit was solved by Euler; it is considered to be the first problem of graph theory [1]. It is easy to design an efficient algorithm to determine whether an Eulerian circuit exists in a graph, and to find one if it does [3, p. 375]. In this paper we consider a modification of the Eulerian circuit problem by restricting circuits to be non-intersecting in a way that is defined below. We prove that the non-intersecting Eulerian circuit problem is NP-complete. This means that the modified problem belongs to a large class of problems (including, for example, the Hamiltonian circuit problem) that are all believed to be impossible to solve efficiently on a computer. (Garey and Johnson give an overview of NP-completeness [2].)

The modification to Euler's problem arises when the edges incident to a vertex are ordered cyclically. Suppose $G = (V, E)$ is a graph for which a "clockwise" order has been defined on the set of edges incident to v, for each vertex v. We say that two edges incident to v are ***neighbors*** at v if they are consecutive in the order. For example, an embedding in the plane of a planar graph G induces such a clockwise order; the neighbors of an edge are the edges that are adjacent to it in some face of the embedding. A path or circuit in G is called ***non-intersecting*** if every two consecutive edges (v_i, v_j) and (v_j, v_k) in it are neighbors at v_j.

The non-intersecting Eulerian circuit problem (NEC for short) asks, given a graph G with clockwise orderings as above, whether G has a non-intersecting Eulerian circuit. The planar non-intersecting Eulerian circuit problem (PNEC) asks, given a planar graph G and an embedding in the plane, whether G has a non-intersecting Eulerian circuit with respect to the ordering induced by the embedding. In general, we are also interested in finding such a circuit if it exists, or, alternatively, in finding a minimal set of pairwise edge-disjoint non-intersecting paths that cover G. However, we show that even the seemingly simple problem of determining existence is computationally very difficult.

The motivation for the definitions above originated from work on flame cutting [5]. Given a stock sheet nested with regular or irregualr parts that need to be cut, for example by a torch. the problem is to optimize the movement of the torch according to several objectives. Minimizing the number of pierce points, or starts, is one important objective. Pierce points usually require materials and operator time for setup. Intersecting paths, although not impossible, should usually be avoided in flame cutting. There are obviously other objectives and restrictions in flame cutting; the entire problem cannot be defined with such a "clean" formulation as above. It is still interesting, however, to identify the difficult parts of the problem. The results of this paper indicate that it is probably computationally infeasible to find a non-intersecting circuit (or path) in an arbitrary planar graph with minimal number of starts.

Another way to view this problem is by considering the graph to be a network of roads. The question is whether one can traverse all the roads without going through the same segment more than once and without crossing an intersection. In this case, if bridges or tunnels are present, then the graph need not be planar.

2. The Main Result

To prove NP-completeness for a problem, we must somehow efficiently encode a known NP-complete problem as an instance of the new problem. The known problem we choose to begin with is SAT, the problem of determining satisfiability of a boolean formula in conjunctive normal form.

Our construction of graphs in which it is difficult to find non-intersecting circuits begins with some facts relating boolean formulas and graphs. To every boolean formula F in conjunctive normal form (CNF), with variables $x_1, \ldots, x_n$, associate a graph $G(F)$. The graph contains one vertex for each variable in F, and one vertex for each clause. The edges of $G(F)$ fall into two classes. The first class of edges forms a cycle through the variable vertices, called the *thread*; it connects the vertices corresponding to variables x_i and x_{i+1} with an edge (for $1 \leq i < n$), and also connects x_n to x_1. The second class of edges connects variable vertices with clause vertices; if variable x_i appears in clause C_j, it connects the corresponding vertices by an edge.

If $G(F)$ is planar, F is called a planar formula. Lichtenstein has shown how to construct in polynomial time, from any formula F in CNF, a planar CNF formula F' and an embedding of $G(F')$ in the plane, such that F' is satisfiable if and only if F is too [4]. We will use Lichtenstein's result as a starting point, but we do not need the details of his construction.

Theorem 1. PNEC *is* NP-*complete.*

Proof: PNEC is clearly in NP; it's easy to guess a path and verify that it has the required properties. To prove completeness, we reduce SAT to PNEC.

Given a formula F in CNF, first use Lichtenstein's construction to obtain a planar formula F' and a planar embedding of $G(F')$. We will alter $G(F')$ to obtain a graph H that has a non-intersecting Eulerian circuit (NEC) if and only if F' is satisfiable, hence if and only if F is satisfiable.

The thread of $G(F')$ divides the plane into two regions, the inside and the outside. A typical variable x is connected to r clauses inside the thread and s clauses outside. Replace x by a *wheel* with $2r + 2s$ vertices on the rim, and one vertex in the center, as in Figure 1. (If $r = 0$, then add two dummy vertices to the rim; similarly if $s = 0$. This guarantees at least one pair of rim vertices lies inside the thread, and one pair lies outside.) Connect each rim vertex to the center by a pair of edges, called a *loop*, and connect neighboring rim vertices by two loops. Label the rim vertices alternately x and $\bar{x}$.

A typical clause C is connected to k variables $x_1, \ldots, x_k$ in clockwise order. (Lichtenstein's construction actually gives $k \leq 3$, but we don't need this fact.) Replace C by a cycle of k vertices, and connect each vertex of this cycle to the rim of the corresponding

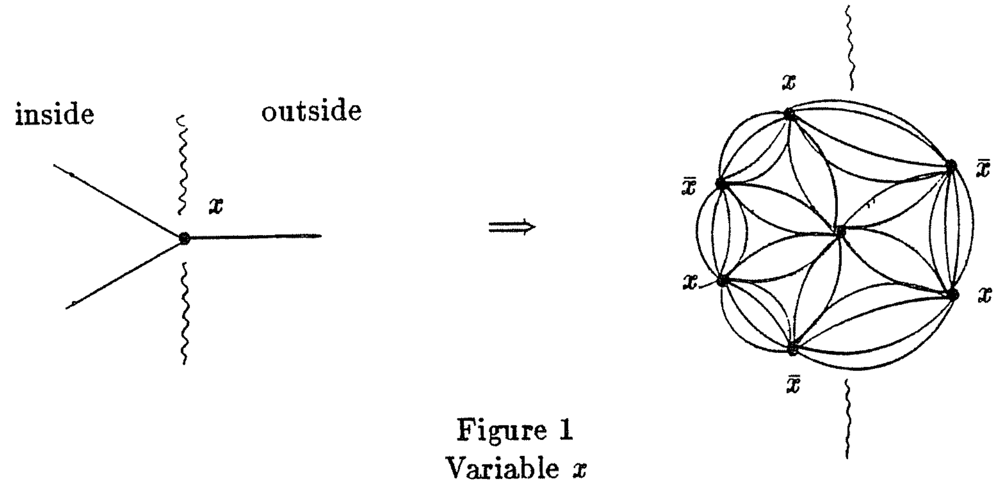

Figure 1
Variable x

variable's wheel by a loop, as in Figure 2. If the literal x_i appears in C, use a rim vertex labelled x_i from x's wheel; if $\bar{x}_i$ appears, use a rim vertex labelled $\bar{x}_i$.

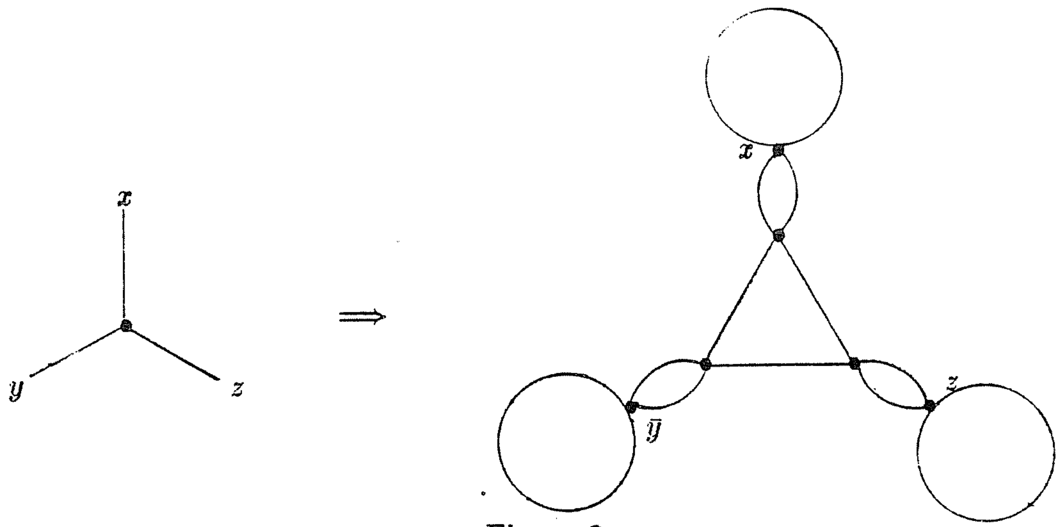

Figure 2
Clause $C = (x \vee \bar{y} \vee z)$

Finally, replace each edge (x_i, x_{i+1}) of the thread by a new vertex, with four loops connecting it to the wheels of x_i and x_{i+1}. Connect two of the loops to x_i's wheel, using the $\bar{x}_i$ vertex just inside the thread and the x_i vertex just outside; connect the other two loops to x_{i+1}'s wheel using an inside x_{i+1} vertex and an outside $\bar{x}_{i+1}$ vertex. The thread edge (x_n, x_1) is handled slightly differently; replace it by three vertices v_s, v_t and v_0, connect v_s to x_1's wheel and v_t to x_n's wheel with two loops apiece, as above, and finally connect v_s and v_t to v_0 by one loop apiece. Figure 3 illustrates this.

We can embed the resulting graph H in the plane in the obvious way, by connecting at most one clause to each pair of rim vertices x and $\bar{x}$, and by letting the loops connecting

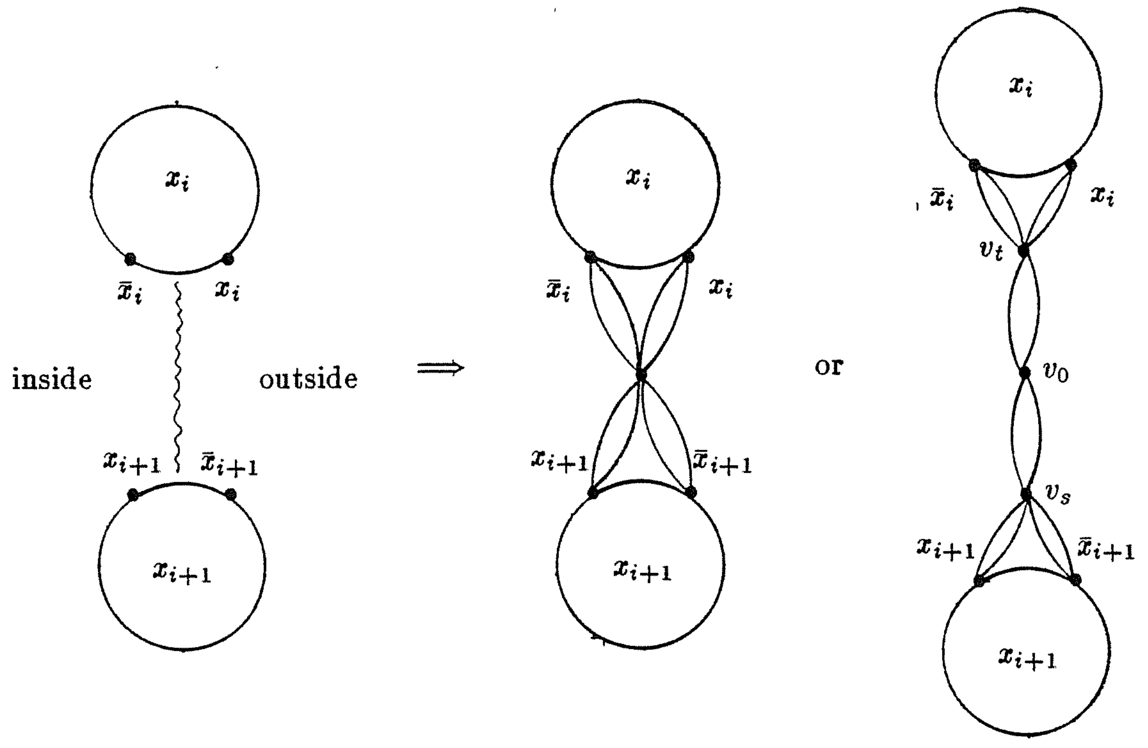

Figure 3
The thread

clauses to variables lie near the corresponding edges of $G(F')$. It remains to show that H has a NEC if and only if F' is satisfiable.

Each wheel or thread vertex v in H has only loops incident to it. So any NEC, upon reaching v along one edge of a loop, must leave v either along the other edge of the loop, or along an edge from a neighboring loop. In the first case, we say the NEC ***rebounds*** along the given loop at v. But any NEC either rebounds along *all* the loops incident to v or rebounds along *none* of them, as in Figure 4. (In the figures, we draw two short lines between edges if the edges appear consecutively in the NEC.) A NEC thus induces a truth assignment to the literals of F', by assigning *true* to a literal at which the NEC never rebounds, and *false* to one at which it always rebounds.

Adjacent rim vertices are labelled x and $\bar{x}$, and are connected by two loops as in Figure 5. A NEC that rebounds at one end of a loop cannot rebound at the other, or else it closes upon itself without using all the edges in the graph. Thus it cannot assign *false* to both literals. Neither can it assign *true* to both literals, since then the two inner edges would form a closed cycle. Thus a NEC assigns *true* to one literal and *false* to the other consistently around the wheel. By a similar argument, a NEC cannot rebound at a center vertex or a thread vertex (except for v_0), since it must rebound at the other end of at least one of the loops incident to the vertex in question.

From Figure 2, we easily see that a NEC in H cannot assign all the literals in a clause the value *false*, since in this case there would be no way for the NEC to escape the

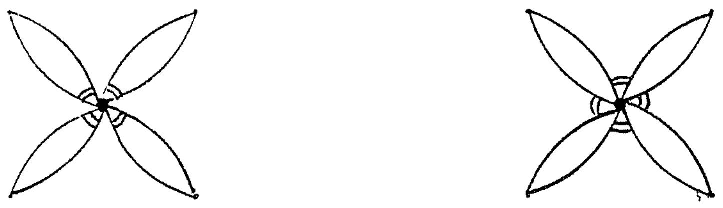

Figure 4
Rebounding at wheel vertices

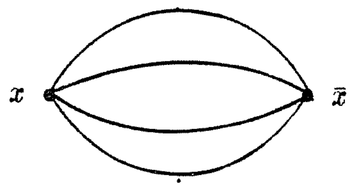

Figure 5
Adjacent rim vertices

subgraph formed by the clause's cycle and its incident loops. This shows that if H has a NEC, then F' must be satisfiable. Conversely, if F' is satisfiable we can find a NEC in H as follows. The NEC leaves v_0 for v_s, traces the inside of the thread until v_t, traces the loop to v_0 and back, traces the outside of the thread back to v_s, and finally returns to v_0. While tracing the wheel of variable x, it proceeds as in Figure 6a if x is *true*, or as in Figure 6b if x is *false*, using the solid edges the first time it reaches the wheel (along the inside) and the dashed edges the second time (along the outside). Note that the NEC can only reach clauses from *true* literals.

Each clause contains at least one true literal. When the NEC reaches the clause's cycle along a true literal's loop for the first time, it traverses the cycle, visiting loops that connect to false literals but avoiding loops that connect to true ones. The latter loops are visited later, when the NEC reaches the wheels belonging to the true literals. The three basic possibilities for a clause with three literals are illustrated in Figure 7. There is always a way for the NEC to proceed in each clause. This completes the proof.

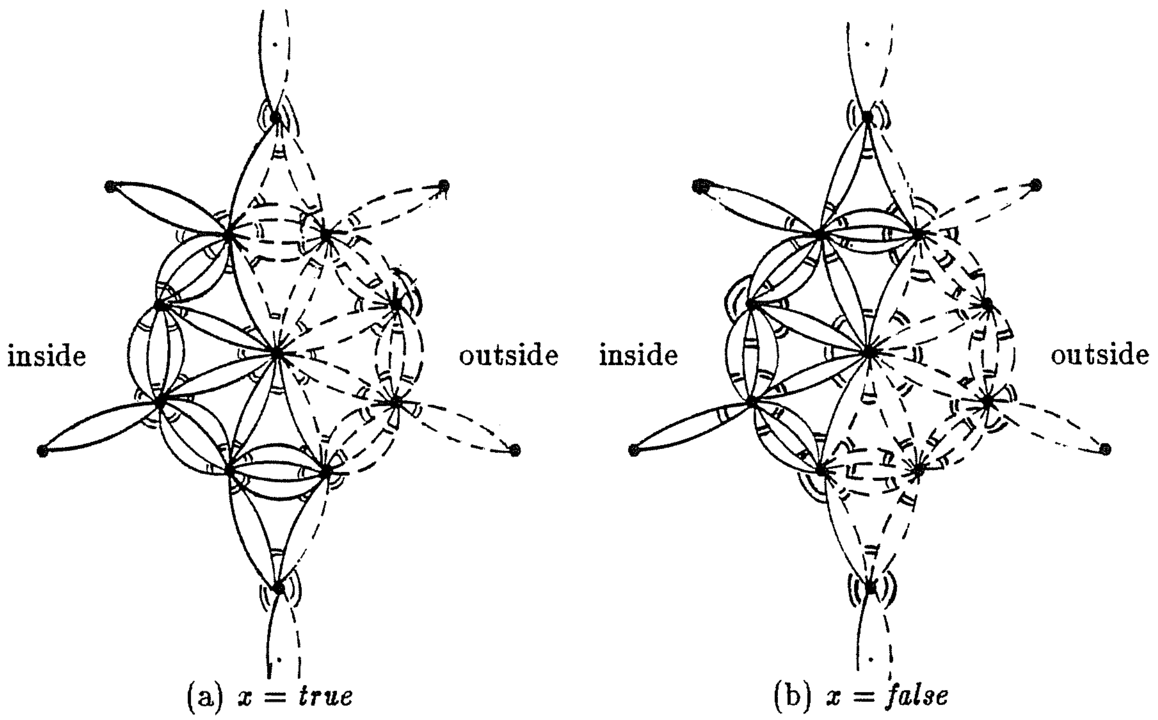

Figure 6
Traversing a wheel

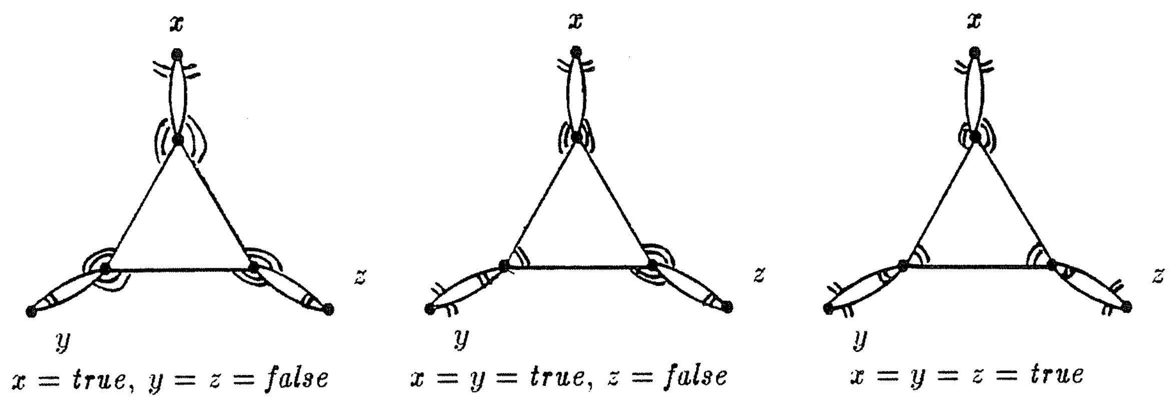

Figure 7
Traversing a clause

3. Extensions

Many variations of the PNEC problem are also NP-complete.

Corollary 2. *It is* NP*-complete to determine whether (a) a directed graph has a non-intersecting Eulerian circuit, (b) a graph or directed graph has a non-intersecting Eulerian path, (c) a graph with no multiple edges has a non-intersecting Eulerian path or circuit.*

Proof: (a) Orient all the loops and cycles in H in the counter-clockwise direction. (b) Attach two new vertices to v_0, lying on the inside of the thread. (c) Introduce vertices of degree two within the edges of H.

Corollary 3. *It is* NP*-complete to determine, for a planar graph G, whether any embedding of G in the plane induces an order for which G has a non-intersecting Eulerian path or circuit.*

Proof: The graph we constructed in Theorem 1 has essentially only one embedding in the plane. To see this, consider a cycle that follows the thread, using the center vertex and two rim vertices from each wheel, as well as the thread vertices. In any embedding, this cycle divides the plane into inside and outside regions; by a suitable inversion of the embedding we can assume these labels correspond to the labels used in Theorem 1.

Near each wheel there are four vertices from the main cycle: the center, two rim vertices, and one adjoining thread vertex. Given any pair of rim vertices not in the main cycle, one from the inside and one from the outside (with respect to the embedding of Theorem 1), it is always possible to find disjoint paths from one of them to the two rim vertices in the cycle, and disjoint paths from the other to the center and thread vertices. Since these paths meet the cycle in an overlapping manner as in Figure 8, they require that the chosen pair of vertices lie on opposite sides of the cycle.

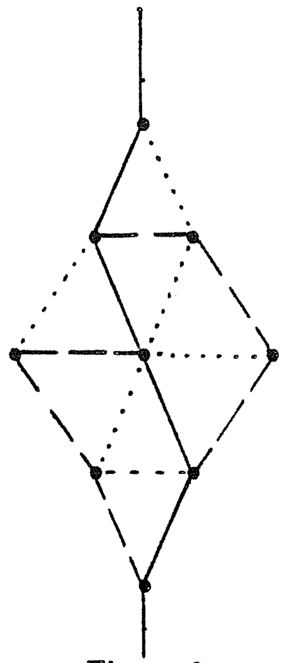
Figure 8
Overlapping paths to the main cycle

Every rim vertex has three disjoint paths to the main cycle, two along the rim and one directly to the center, so no rim vertex can be embedded within a region bounded by a loop. Thus, the only choice in embedding a wheel is whether to interchange the inside and outside portions of the rim.

No clause can be embedded within the region bounded by a wheel, provided that the clause uses at least two different variables. Nor can a clause be embedded within a loop. Thus the graph must be embedded essentially in the way described in Theorem 1, except that clauses can be moved from the inside to the outside of the thread or vice-versa. It is also possible for one clause to completely encircle the entire graph, with the loops leading to variables lying on the inside of its cycle. None of these alterations affect the proof of Theorem 1. This proves the corollary.

References

[1] Leonhard Euler, "Solutio problematis ad geometriam situs pertinentis". *Commentarii Academiae Scientiarum Petropolitanae* **8** (1736), 128–140. Reproduced in *Leonhardi Euleri Opera Omnia, Series I* **7** (1923), 1–10.

[2] Michael R. Garey and David S. Johnson, *Computers and Intractability: A Guide to NP-Completeness.* W. H. Freeman, San Francisco (1979).

[3] Donald E. Knuth, *The Art of Computer Programming.* Volume I: *Fundamental Algorithms.* Addison-Wesley, Reading, MA (second edition, 1973).

[4] David Lichtenstein, "Planar formulae and their uses". *SIAM Journal on Computing* **11** (1982), 329–343.

[5] Udi Manber and Sharat Israni, "Pierce point minimization and optimal torch path determination in flame cutting". *Journal of Manufacturing Systems* (to appear).