FINDING PSEUDOPERIPHERAL NODES IN GRAPHS

Jan K. Pachl

Department of Computer Science
University of Waterloo
Waterloo, Ontario
Canada N2L 3G1

Research Report CS-82-20

June 1982

This research was supported by the Natural Sciences and Engineering Research Council of Canada under grant A7403.

1. Overview.

SPARSPAK, Waterloo Sparse Linear Equations Package, contains a subroutine called Pseudoperipheral Node Finder, whose goal is to find a node with large eccentricity in a given sparse graph [4] [5]. In their book, George and Liu ask whether the execution time of the subroutine can be worse than linear in the number of edges ([5], p. 75).

This paper answers the question: the *worst case* execution time of the subroutine on graphs with n nodes and e edges is at least $\Omega(e\sqrt{n})$. No upper bound of the same order seems to be known for the SPARSPAK algorithm, but there is another algorithm for finding pseudoperipheral nodes, whose worst case execution time is $O(e\sqrt{n})$.

2. The SPARSPAK pseudoperipheral node finder.

Let $G=(X,E)$ be a graph with the set X of nodes and the set E of edges. Assume that for every two nodes $x,y \in X$ there is a path from x to y; the length of the shortest such path is called the *distance* between x and y and denoted $d(x,y)$. The *eccentricity* of $x \in X$ is defined by

$$l(x) = \max \{ d(x,y) \mid y \in X \} ,$$

and the *diameter* of G by

$$\delta(G) = \max \{ l(x) \mid x \in X \} = \max \{ d(x,y) \mid x,y \in X \} .$$

A node $x \in X$ is called *peripheral* if $l(x)=\delta(G)$.

Experience shows that several node ordering algorithms used in sparse matrix computations perform well when their starting nodes have large eccentricity. Peripheral nodes are expensive to find; the best algorithms known have time complexity $O(M(n)\log n)$ for dense graphs [2] and $O(ne)$ for sparse ones [3]. SPARSPAK uses pseudoperipheral nodes instead. We say that $x \in X$ is a *pseudoperipheral node* if there exists $y \in X$ such that

$$l(x) = d(x,y) = l(y).$$

The term is used in a different meaning in [4], where $x \in X$ is said to be pseudoperipheral if $l(x)$ is "close" to $\delta(G)$. The present terminology is less vague, and it remains consistent: the pseudoperipheral node finder indeed finds a pseudoperipheral node.

The following description of the SPARSPAK pseudoperipheral node finder employs a function *Furthest_from(x)*, which returns $y \in X$ such that $d(x,y)=l(x)$; if there are several such y then one is selected arbitrarily. This is the algorithm:

```
x_0 := any element of X
j := 0
x_1 := Furthest_from(x_0)
repeat
    j := j+1
    x_{j+1} := Furthest_from(x_j)
until  d(x_{j+1},x_j) = d(x_j,x_{j-1})
claim  x_j  is  pseudoperipheral
```

We first consider the question of how many times the algorithm calls the function *Furthest_from*.

2.1. Theorem. *If $w(n)$ denotes the worst case number of calls to Furthest_from by the SPARSPAK pseudoperipheral node finder on graphs with n nodes, then*

$$w(n) \geq \Omega(\sqrt{n}) .$$

Proof. There is a sequence of graphs G_1, G_2, ..., such that for each $k = 1,2,...$

(i) G_k has $n = k^2+9k+3$ nodes and n edges;

(ii) there is a node x_0 of G_k such that the pseudoperipheral node finder starting at x_0 calls the function *Furthest_from* $2k+1$ times.

Figs. 1 and 2 show two graphs in the sequence, G_2 and G_3. For a general $k \geq 1$, the graph G_k consists of a cycle whose nodes are, consecutively, y_0, y_1, ..., y_{6k+1}, and linear segments attached to certain nodes in the cycle. A segment of length s is attached to the node y_j if and only if either $j = 2i$, $s = i+1$ and $0 \leq i \leq k$, or $j = 3k+2i$, $s = i+1$ and $1 \leq i \leq k$.

From (i) and (ii) it follows that on a graph with $n = k^2+9k+3$ nodes and n edges the algorithm makes $2\sqrt{n + \frac{69}{4}} - 8 = 2\sqrt{n} + O(1)$ calls to the function.

□

2.2. Conjecture. *There is a constant c such that, for every graph on n nodes and for every starting node x_0, the pseudoperipheral node finder calls the function Furthest_from at most $c\sqrt{n}$ times.*

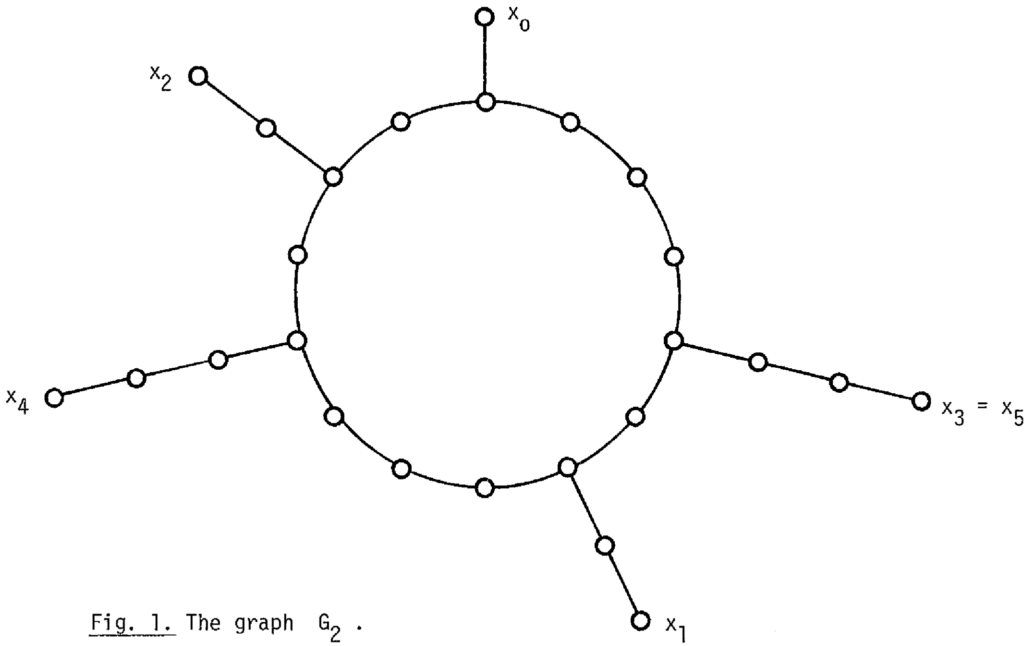

Fig. 1. The graph G_2 .

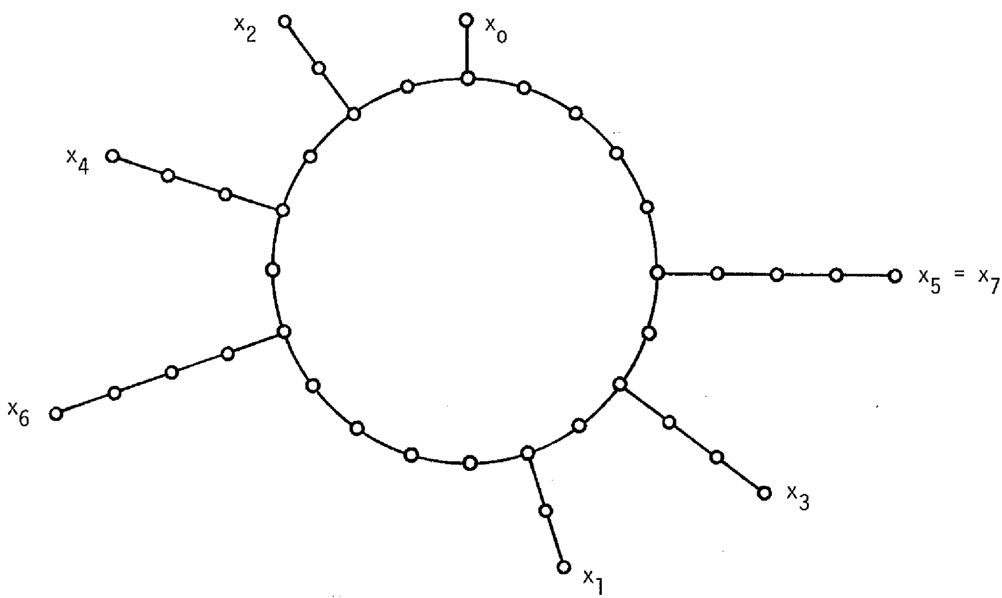

Fig. 2. The graph G_3 .

3. The worst case execution time.

In SPARSPAK, the node $y = Furthest\_from(x)$ is computed by the breadth first search ([3], p. 12). The graph is represented by its incidence lists ([3], p. 4). If we assume the uniform cost criterion ([1], 1.3) then one call to *Furthest_from* requires time proportional to e, the number of edges.

Hence the conjecture in section 2 states that the worst case execution time of the SPARSPAK pseudoperipheral node finder for the graphs with n nodes and e edges is $O(e\sqrt{n})$; and from 2.1 it follows that $\Omega(e\sqrt{n})$ is a lower bound for the algorithm.

Although we do not know whether the complexity of the ***algorithm*** is really $O(e\sqrt{n})$, we are now going to see that the complexity of the ***problem*** is not worse than $O(e\sqrt{n})$.

Let $G=(X,E)$ be a graph with n nodes and e edges, and let k be a positive integer. We say that a set $Y \subseteq X$ is *k-discrete* if $d(x,y) > k$ whenever $x,y \in Y$, $x \neq y$.

3.1. Lemma. *There is an algorithm that constructs a maximal k-discrete set of nodes and whose worst case execution time is O(e).*

Proof. Denote

$$B_k(x) = \{ \ y \in X \ | \ d(x,y) \leq k \ \} \ .$$

If $B_k(x)$ is computed by the breadth first search, then the following algorithm accesses no edge more than twice and its worst case execution time is $O(e)$.

```
S := ϕ
repeat
    x := any element of X
    S := S ∪ {x}
    X := X − B_k(x)
until X = ϕ
claim S is a maximal k−discrete set
```

□

3.2. Lemma. *If $n \geq k/2$ then every k-discrete set $Y \subseteq X$ has at most $2n/k$ nodes.*

Proof. Denote $h = \lfloor k/2 \rfloor$. The sets $B_h(x)$ and $B_h(y)$ are disjoint when $x,y \in Y$, $x \neq y$. Moreover, if $n \geq k/2$ then every $B_h(x)$ has at least $k/2$ elements (because G is connected). Hence the cardinality of Y is at most $\frac{n}{k/2} = \frac{2n}{k}$.

□

3.3. Lemma. *There is an algorithm to find, for every $Y \subseteq X$, two nodes $x_0, y_0 \in Y$ such that*

$$d(x_0, y_0) = \max \{ d(x,y) \mid x,y \in Y \} ;$$

the worst case execution time of the algorithm is O(me), where m is the cardinality of Y.

Proof. All distances $d(x,y)$ for a given x can be computed by the breadth first search starting at x, which requires time $O(e)$. Therefore all the distances $d(x,y)$, $x,y \in Y$, can be computed in time $O(me)$.

□

We are ready to construct the $O(e\sqrt{n})$ algorithm for finding pseudoperipheral nodes.

3.4. Theorem. *There is an algorithm that finds a pseudoperipheral node in worst case time* $O(e\sqrt{n})$.

Proof. Let $k=\lceil\sqrt{n}\rceil$. The algorithm has three parts:

1. Find a maximal k-discrete set $Y\subseteq X$.

2. Find $x_0,y_0\in Y$ such that

$$d(x_0,y_0) = \max\{\, d(x,y) \mid x,y\in Y \,\} .$$

3. Execute the pseudoperipheral node finder of section 2 with starting node x_0.

By 3.1, 3.2 and 3.3, steps 1 and 2 can be executed in worst case time $O(e)$ and $O(e\sqrt{n})$, respectively. To estimate the execution time of step 3, observe that for any two nodes $x,y\in X$ there are $x',y'\in Y$ such that $d(x,x')\le k$ and $d(y,y')\le k$ (because Y is maximal k-discrete). Therefore

$$l(x_0) \ge d(x_0,y_0) \ge \delta(G)-2k .$$

The sequence $x_0,x_1,\cdots$ generated by the pseudoperipheral node finder satisfies

$$\delta(G)-2k \le l(x_0) < l(x_1) < \cdots \le \delta(G) .$$

Hence the pseudoperipheral node finder in step 3 repeats its loop at most $2k$ times. It follows that the worst case execution time for step 3 is $O(e\sqrt{n})$.

□

4. Concluding remarks.

The *worst case* time cost of the algorithm in section 3 is $O(e\sqrt{n})$, which is not worse than the *worst case* time cost of the SPARSPAK pseudoperipheral node finder. Nevertheless, the SPARSPAK algorithm seems to execute in time $O(e)$ on "typical" graphs arising in sparse matrix computations, and is therefore better in practice.

The space cost of both algorithms is dominated by the memory needed to store the graph; it is proportional to $n+e$.

Acknowledgements. I wish to thank K. Booth, A. George and D. Rotem for their comments and helpful suggestions.

References.

[1] A. V. Aho, J. E. Hopcroft and J. D. Ullman: The Design and Analysis of Computer Algorithms, Addison-Wesley (1974).

[2] K. S. Booth and R. J. Lipton: Computing extremal and approximate distances in graphs having unit cost edges, Acta Informatica 15 (1981), 319-328.

[3] S. Even: Graph Algorithms, Computer Science Press (1979).

[4] A. George and J. W. H. Liu: An implementation of a pseudoperipheral node finder, ACM Trans. Math. Software 5 (1979), 284-295.

[5] A. George and J. W. H. Liu: Computer Solution of Large Sparse Positive Definite Systems, Prentice-Hall (1981).