Note

An efficient algorithm for finding a two-pair, and its applications

Srinivasa R. Arikati* and C. Pandu Rangan
Department of Computer Science and Engineering, Indian Institute of Technology, Madras 600 036, India

Received 11 July 1989
Revised 23 July 1990

Let \( G = (V, E) \) be an undirected, finite, and simple graph. All the definitions not given here may be found in [1]. A pair \((u, v)\) of nonadjacent vertices in \( G \) form a two-pair if every chordless path between \( u \) and \( v \) is of length two.

We shall digress briefly to remark on the applications of our problem. One of the fundamental notions in computational geometry is the concept of visibility. A visibility graph has vertices which correspond to geometric components such as points or lines, and edges which correspond to visibility of these components from each other. O'Rourke [5] declares that the fundamental problems involving visibility in computational geometry will not be solved until the combinatorial structure of the visibility graphs is more fully understood. There is an intimate connection between the visibility problem and the polygon covering problem. Recently, several interesting structural characterizations of visibility graphs for regions inside a simple orthogonal polygon were reported in [3, 4]. In particular, the visibility graph of a special type of orthogonal polygon called Class 3 polygon [3] is a weakly triangulated graph. It is also shown in [3, 4] that the polygon covering problem for the Class 3 polygon reduces to the clique covering problem on weakly triangulated graphs. The fact that weakly triangulated graphs are perfect is used to derive several insightful duality relationships for Class 3 polygon.

* Currently, with Department of Mathematics and Computer Science, The University of Illinois at Chicago, Box 4348, Chicago, IL 60680, USA.
The minimum clique cover problem for a weakly triangulated graph can be solved in $O(n^5)$ time [2]. In fact, $O(n^5)$ algorithms for the maximum clique problem, maximum independent set problem, and the minimum coloring problems on weakly triangulated graphs are also presented in [2]. For all these problems finding a two-pair is a crucial subproblem. In fact, it is shown in [2] that if $T$ is the time required to find a two-pair, then all the above four problems can be solved in $O(n \cdot T)$ time. The earlier algorithm [2] has $T = O(n^7 \cdot m)$. In this note, we improve this time to $O(n \cdot m)$. Hence, our algorithm cuts down the run time of every problem mentioned above by a factor of $n$.

From now on, let us assume that $v$ is a fixed vertex in $G$, and $N(v) = \{u | u \text{ is adjacent to } v\}$. Let the connected components of $H = G - \{v\} - N(v)$ be $H_1, \ldots, H_k$. Our algorithm consists of identifying a vertex in $H$ that can form a two-pair with $v$.

For $1 \leq i \leq k$, define $N_i(v) = \{u | u \in N(v) \text{ and } u \text{ is adjacent to some vertex in } H_i\}$.

**Lemma 1.** Let $u$ be any vertex in some connected component, say $H_i$, of $H$. Then, $(v, u)$ is a two-pair if and only if $u$ is adjacent to every vertex of $N_i(v)$.

**Proof.** The proof is clear. $\square$

For every vertex $u \in H$, define $\text{label}(u)$ to be the index such that $u \in H_{\text{label}(u)}$. Also, for every vertex $u \in H$, define $N_{\text{label}(u)}(v) = \{w | w \text{ is adjacent to } u \text{ and } w \in N(v)\}$.

**Lemma 2.** A pair $(v, u)$ of vertices, where $u \in H$, form a two-pair if and only if $|N_{\text{label}(v)}(v)| = |N_{\text{label}(u)}(v)|$.

**Proof.** From Lemma 1, it is clear that $(v, u)$ is a two-pair if and only if $N_{\text{label}(v)}(v) = N_{\text{label}(u)}(v)$. It follows from the definitions of $N_{\text{label}(v)}(v)$ and $N_{\text{label}(u)}(v)$ that $N_{\text{label}(v)}(v) = N_{\text{label}(u)}(v)$ if and only if $|N_{\text{label}(v)}(v)| = |N_{\text{label}(u)}(v)|$. $\square$

We now present the algorithm below.

**Algorithm** \textsc{two-pair}

**Input:** A graph $G = (V, E)$, and adjacency lists denoted by $N(v), v \in V$.

**Output:** A two-pair, if it exists.

**Method:**

1. for all $v \in V$ do
   
   begin
   
   1.1. Compute the connected components $H_1, \ldots, H_k$ of $H = G - \{v\} - N(v)$;
   
   1.2. for all $x \in N(v)$ do label($x$) := 0;
   
   1.3. for all $u \in H$ do compute label($u$);
   
   1.4. for all $u \in H$ do compute $|N_{\text{label}(u)}(v)|$;
An efficient algorithm for finding a two-pair

Step 1.5. (*compute $N_i(u)$, $1 \leq i \leq k$*)

for $i := 1$ to $k$ do $N_i(u) := \emptyset$;

for all $x \in N(u)$ do

begin

for all $u \in N(x)$ do

if $\text{label}(u) \neq 0$ (*$u \in H$*)

then $N_{\text{label}(u)}(v) := N_{\text{label}(u)}(v) \cup \{x\}$

(*This is not a disjoint union, but can be done in constant time*)

end;

Step 1.6. for all $u \in H$ do compute $|N_{\text{label}(u)}(v)|$;

Step 1.7. for all $u \in H$ do

if $|N_{\text{label}(u)}(v)| = |\text{NV}(u)|$

then declare $(u, u)$ is a two-pair and STOP;

end

end TWO-PAIR.

Theorem. Algorithm TWO-PAIR finds a two-pair, if it exists, in a graph $G = (V, E)$ in $O(n \cdot m)$ time, where $n = |V|$, $m = |E|$.

Proof. Correctness follows from Lemma 2. To analyse the run time, we note that Steps 1.1-1.7 each take $O(m)$ time. Hence, each iteration of Step 1 takes $O(m)$ time. This implies that the total run time is $O(n \cdot m)$. □

Conclusion. Our algorithm for two pair improves the complexity of a number of algorithms by a factor of $n$. We find a two-pair in an arbitrary graph in $O(n \cdot m)$ time. We strongly feel that faster algorithms can be devised for finding a two-pair if we restrict the input graph to be a weakly triangulated graph. Such an algorithm will yield further reduction in the complexity of all the problems we have mentioned earlier in this paper.

Acknowledgement

We thank the referees for their helpful comments.

References

