**South African Computer Journal**
Number 23
July 1999

**Suid-Afrikaanse Rekenaar-tydskrif**
Nommer 23
Julie 1999

# Computer Science and Information Systems

# Rekenaarwetenskap en Inligtingstelsels

**The South African Computer Journal**
*An official publication of the Computer Society of South Africa and the South African Institute of Computer Scientists*

**Die Suid-Afrikaanse Rekenaartydskrif**
*'n Amptelike publikasie van die Rekenaarvereniging van Suid-Afrika en die Suid-Afrikaanse Instituut vir Rekenaarwetenskaplikes*

**World-Wide Web:** `http://www.cs.up.ac.za/sacj/`



**Subscriptions**

| | Annual | Single copy |
|---|---|---|
| Southern Africa | R80.00 | R40.00 |
| Elsewhere | US$40.00 | US$20.00 |

An additional US$15 per year is charged for airmail outside Southern Africa

to be sent to:

*Computer Society of South Africa*
*Box 1714, Halfway House, 1685*
*Phone: +27 (11) 315-1319 Fax: +27 (11) 315-2276*

# Guest Editorial
# Computer Science and Information Systems: The Future?

Philip Machanick

*Department of Computer Science, University of the Witwatersrand, South Africa*
philip@cs.wits.ac.za

## 1 Introduction

As president of the South African Institute for Computer Scientists and Information Technologists (SAICSIT), I have visited a number of campuses and companies, in an attempt at arriving at a general assessment of the state of our subjects in South Africa.

An issue which I consistently pick up is that while everyone seems to think that computer-related skills are extremely important and in short supply, our academic departments are also extremely under-resourced.

At the last Southern African Computer Lecturers Association (SACLA) conference (28-29 June, Golden Gate), I had the opportunity to discuss the problems other academics see. This editorial lists some of the problems reported at SACLA, and proposes a way forward.

## 2 Problems

At SACLA, I led a discussion of problems seen in our academic departments.

There was wide agreement that both Computer Science (CS) and Information Systems (IS) departments were under pressure to increase student numbers (massification), and were seen as cash cows to prop up less popular subjects. It was broadly agreed that staffing was a critical issue: too few posts for the workload, salaries way out of line with industry (half or less, as compared to the US, where an academic salary may be 80% of an industry salary). Recent graduates often make more than professors which makes it hard to persuade our students to become academics (even to do higher degrees). Attracting a recent PhD with a sense of adventure is may be possible, but attracting experienced people used to earning a salary in a strong currency is hard. IS jobs are worse than CS, as the skills required are more like those in business. Support staff salaries are an even harder issue: their skills relate even more directly to job descriptions in industry.

A problem in addressing our concerns is that we are so overworked that we don't have time for "politics": academics with no students have time on their hands, but we don't. More industry support not only with directly addressing problems but with taking on university administrations would be useful, but they too have major problems and don't have free time.

## 3 Solutions?

Solutions are harder to identify than problems.

The SACLA session ended with a proposal that we conduct surveys of our institutions and businesses, to find out what the problems are, as a starting point for going to university administrations, government and business.

Another idea was to attempt to find common cause with business in taking on problems they have in common with academia, including the skills shortage, the insufficient capacity of our education system, and dealing with employment equity.

One of our biggest difficulties is to free up time to deal with issues such as resource allocation within our universities. The "competition" is frequently other academics with time on their hands, since they have too few students, and therefore are in a position to spend time looking after their interests.

What is needed now is some thought about how to pull ourselves out of the mess we are in. In particular, we need strategies to exploit our strengths: our high demand among students, the high demand for the skills we produce and the ubiquitous applicability of computer technology.

Given the wide use of computers, it would seem obvious that our areas should be strongly supported by a range of role players, yet the fact that so many different groups are interested in computer technology in one way or another has tended to fragment efforts to enhance our industry and academic institutions.

Clearly, from conversations I have held, some departments are in much better shape than others. Even so, some kind of collective effort is likely to achieve more results than if we allow ourselves to be pushed around as individuals. Addressing the fragmentation of efforts seems a worthy goal in itself, to reduce duplication and contradictory goals.

I appeal to anyone who has constructive ideas on how to take our subjects forward to contact me. Let us work on building ourselves up. The economy depends on us, much more than on most other academic disciplines. It's time we made that point, and made it strongly.

# SAICSIT'99

South African Institute of Computer Scientists and Information Technologists

**Annual Research Conference 17-19 November 1999**

## Prepare for the New Millennium

***Is there life after y2k?***

## Mount Amanzi Lodge, Hartebeespoort

near Johannesburg and Pretoria

## keynote speaker: Barbara Simons, ACM President

and many other local and international speakers (academic and industry)

## Call for Participation

papers in a many areas of Computer Science and Information Systems are expected

Price Waterhouse Coopers prizes: Best Paper R10000 • Best Student Paper R5000

*please check the conference web site for accepted papers:*

***http://www.cs.wits.ac.za/~philip/SAICSIT/SAICSIT-99/***

## To Register

*go to the conference registration web page:*

***http://www.cs.wits.ac.za/~philip/SAICSIT/SAICSIT-99/reservation.html***

*or contact* **SAICSIT'99 Secretary** *for details:*

**Department of Computer Science, Senate House 1137**
**University of the Witwatersrand**
**Jorissen Street**
**Wits, 2050**
**South Africa**

phone (011)716-3309 fax 339-3513 (international: replace 011 by 27-11)

*saicsit99-info@cs.wits.ac.za*

## Dates and Publication Details

early booking deadline: 14 September • on-site registration starts: 17 November 1999

workshops, tutorials 17 November 1999 • paper sessions: 18-19 November 1999

*papers will appear in a special issue of* South African Computer Journal

## sponsors

NRF PRICEWATERHOUSECOOPERS

# Orthogonal Ray Guarding of Adjacencies between Orthogonal Rectangles

Ian Sanders[a], David Lubinsky[b], Michael Sears[c] and Derrick Kourie[d]

[a]*Department of Computer Science, University of the Witwatersrand, Johannesburg,* ian@cs.wits.ac.za
[b]*Department of Computer Science, University of the Witwatersrand, Johannesburg,* [c]*Department of Computational and Applied Mathematics, University of the Witwatersrand, Johannesburg,* sears@gauss.cam.wits.ac.za
[d]*Department of Computer Science, University of Pretoria,* dkourie@cs.up.ac.za

## Abstract

*Guarding and covering problems have great importance in Computational Geometry. In this article the notion of a* ray guard, *a guard that can only 'see' along a single ray, is introduced. The problem of siting the fewest possible such guards so that they guard all adjacencies in an orthogonal arrangement of adjacent non-overlapping rectangles is discussed. The problem is further restricted by requiring that the direction of sight be parallel to an axis and that the guards cannot 'see' outside the rectangles. The problem is motivated by applications in architecture and urban planning. This article shows that the problem is NP-Complete because of the locally indeterminate choice which can be introduced in positioning guards.*

**Keywords:** *Computational Geometry, NP-Complete, Covering, Guarding, Orthogonal Rectangles*
**Computing Review Categories:** *F.2, F.2.2 (Also G.2.1 and G.2.2)*

## 1 The Problem

### 1.1 Background

Guarding and covering problems are common in the field of Computational Geometry (see O'Rourke's monograph [19], the survey papers by Shermer [22] and Urrutia [25] and the summaries of results by O'Rourke [20] and Suri [24]). This article introduces a new variation on guarding problems where the guards can only see along a single ray. The problem has its origin in the area of town planning and urban design — Hillier *et al*'s idea of *Space Syntax Analysis* [16]. The idea of Space Syntax Analysis is to give a globalising perspective of the design by determining how easy it is to traverse the town. This analysis is accomplished by the positioning of axial lines on a town plan — the fewest such lines are required.

The problem is similar to the many guarding problems [3, 12, 5, 9] since the lines can be thought of as guards whose vision is restricted to a single ray. The situation can be envisaged as an art gallery made up of a number of adjacent rooms where the designers wish to position the most doors between rooms (to allow easy access) in such a fashion that all doorways can be guarded by the minimum number of ray guards.

Another application of this problem is in the design of integrated circuits. Here the problem is the siting of the fewest connecting strips to join all of the components on the chip.

This article considers the problem where the rays are restricted to being parallel to the Euclidean axes and the space cut by the rays is a collection of adjacent orthogonal rectangles. The problem is presented below — Section 1.2.

### 1.2 Statement of the Problem

Given a number of adjacent, orthogonally-aligned rectangles find the fewest orthogonal line segments, contained wholly inside the rectangles, required to cut all of the boundaries shared between adjacent rectangles. An additional requirement is that each line segment should cut as many of the shared boundaries as possible.

The solutions for horizontal line segments (and vertical shared boundaries) and vertical line segments (and horizontal shared boundaries) are independent and the remainder of the article will only discuss the former.

Depending on how the problem is considered there are 2 similar but distinct problems which can be addressed.

1. The shared boundaries between adjacent rectangles can be crossed more than once but every shared boundary must be cut at least once.

2. Any shared boundary between adjacent rectangles has exactly one orthogonal line segment passing through it.

Figure 2 shows the difference between these two specifications for a simple configuration of adjacent rectangles. In problem 1 the leftmost adjacency is cut by lines $a$, $c$ and $d$. In problem 2, any of $a'$, $c'$ or $d'$ could have cut the leftmost adjacency but only $d'$ actually does. In this article only problem 1 is addressed.

### 1.3 Related Work

The problem of covering a polygon with simpler polygons has been the subject of much research [27, 10, 7, 8] and many of the problems in this class have been shown to be NP-Hard. One of these problems is that of finding the minimum number of star polygons needed to cover a given polygon. This problem is equivalent to the placement of the minimum number of point guards so that each point inside the polygon is visible to some guard. This problem is intractable for polygons with holes and remains so for simple polygons. Other guarding problems are discussed by Györi et al [14], Shermer [23], Bjorling-Sachs [2] and Bjorling-Sachs and Souvaine [4, 5] among others.

Lee and Lin [17] discuss the computational complexity of the art gallery guarding problem. They prove that the problem for a simply connected simple polygon is NP-Hard for the *minimum vertex guard* (where guards must be located at the vertices of the polygon), *minimum point guard* (where guards can be placed anywhere in the interior of the polygon or on its boundary) and *minimum edge guard* (where guards can only be placed on the edges which make up the polygon boundary and are allowed to move along the edge on which they are placed) versions of the problem. Their proof is based on a transformation from Boolean Three Satisfiability (3SAT).

Section 4 presents a proof which shows that the ray guarding problem introduced in Section 1.2 is NP-Complete.

### 1.4 The layout of the article

The next section of the article defines the terminology used. Section 3 discusses the problem in somewhat more detail and introduces the issue of choice in placing the guards. Section 4 shows that the problem is NP-Complete by means of a transformation from vertex cover and Section 5 presents some cases where the problem can be solved in polynomial time. Some ideas for future research are presented in Section 6.

## 2 Terminology

In the remainder of this article the common definitions in Computational Geometry and Graph Theory are used. The definitions given below have particular relevance to this article.

- A line is orthogonal, or orthogonally aligned, if it is parallel to one of the Cartesian axes.
- If all of the edges of a rectangle are orthogonally aligned then this rectangle is refered to as an orthogonal rectangle.
- Edges of rectangles (or portions of edges) which are coincident are called adjacencies. Only vertical adjacencies (horizontal lines) are considered in this article — the problem for horizontal adjacencies is similar.
- A ray is defined by the set of adjacencies between rectangles which can be cut by an orthogonal line segment through these adjacencies.
- A ray must be wholly contained in the collection of rectangles. An example of a valid ray is $a$ in Figure 2 — it cuts the adjacencies between rectangles 1 and 2, 2 and 5 and 5 and 7.
- A maximal ray is a ray such that if any other ray includes the same set of adjacencies then it includes no other adjacencies. Ray $a$ in Figure 2 is a maximal ray. Any ray which only cuts the adjacencies between rectangles 1 and 2 and 2 and 5 would not be maximal.
- A planar graph is a graph which can be drawn or embedded in the plane in such a way that the edges of the embedding intersect only at the vertices of the graph.
- A cut vertex is a vertex whose removal increases the number of components in a graph.
- A biconnected graph is a graph which contains no cut vertices.
- A biconnected planar graph is a planar graph which contains no cut vertices.
- A face in a planar representation of a graph is a planar region bounded by edges and vertices of the representation and containing no edges or vertices in its interior.

Note that throughout this document any mention of rectangles should be taken to mean orthogonal rectangles.

## 3 Addressing the problem

At first glance this problem would appear to be easy to solve. Given $n$ rectangles, the upper bound on the number of possible adjacencies is of $O(n)$ and a simple lower bound for finding the adjacencies can be shown to be $\Omega(n \log n)$.

The upper bound can be easily shown by reducing the rectangles and their adjacencies to the form of a graph where the nodes in the graph represent the rectangles and the edges of the graph represent adjacencies between two rectangles. The graph generated in this way must be planar, thus the maximum number of edges (adjacencies) can be determined from Euler's formula which gives $e \leq 3v - 6$ ($e$ the number of edges and $v$ the number of vertices). This implies that the number of adjacencies must be $O(n)$.

The lower bound follows by a transformation from ***element uniqueness*** — given $N$ real numbers, decide if any two are equal — which has an $\Omega(n \log n)$ lower bound [21]. Given $N$ real numbers $\{z_1, \ldots, z_N\}$ and an interval $[b,t]$ then for each $z_i$ construct a rectangle defined by bottom left corner $(b, z_i)$ and top right corner $(t, z_i)$. Determining whether any two rectangles are adjacent is now the same as determining ***element uniqueness***.

The question is then: How easy is it to find the minimum number of maximal rays which guard all of the adjacencies?

The problem is interesting and difficult to solve efficiently because of the issue of choice. The simplest case of choice is illustrated in Figure 3. In this case there are seven adjacencies which must be crossed by rays. It is easy to see that most of the adjacencies can be cut by the rays marked *a* and *b* (0-3-4-6 and 2-3-5-6) but the adjacency between rectangles 1 and 3 can be cut by rays *c* (1-3-4-6) and *d* (1-3-5-6). Only one of these "choice" rays is actually necessary. More complicated choice situations can arise as the number of rectangles to be considered grows. An algorithm to solve the problem must be able to resolve conflicts of this type.

# 4 Proving NP-Completeness of the problem of resolving choice

This section shows the problem of finding the minimum number of maximal rays to guard all of the adjacencies in a configuration of adjacent rectangles is NP-Complete. The proof of this will be accomplished through a transformation from ***vertex cover*** for a planar graph [11, 18], to a restricted instance of the problem under consideration, i.e. the problem of choosing the fewest maximal rays to guard all the adjacencies in a collection of rectangles.

Planar vertex cover is defined as

***planar vertex cover***
*Instance:* Planar graph $G = (V,E)$, positive integer $K \leq |V|$.
*Question:* Is there a vertex cover of size $K$ or less for $G$, i.e. a subset $V' \subseteq V$ with $|V'| \leq K$ such that for each edge $\{u,v\} \in E$ at least one of $u$ and $v$ belongs to $V'$?

and the ray guarding problem can be stated as

***ray guard***
*Instance:* A collection of orthogonal rectangles $R_1 \ldots R_n$, where each $R_i$ is adjacent to at least one other rectangle, and a positive integer $O \leq 4n$.
*Question:* Is there a set $P$ of rays where each ray is maximal, each vertical adjacency is crossed at least once by the rays in $P$ and $|P| \leq O$?

The transformation from ***planar vertex cover*** [11, 18] will be done by mapping vertices in a planar graph to choice rays in the problem being considered. Edges in the planar graph will be mapped to adjacencies which are guarded by the choice rays. In this mapping an edge between two vertices represents an adjacency which is guarded by two choice rays.

This transformation will be done in two steps. First, a planar graph is transformed to a 'stick diagram'. In this 'stick diagram' each vertex in the original graph is mapped to a horizontal line representing a choice ray and each edge in the original graph is mapped to a vertical line which is cut by the two horizontal lines which represent the two vertices to which the edge is incident. The problem then becomes that of choosing the minimum number of horizontal lines to cut all of the vertical lines.

***stick diagram***
*Instance:* A collection $H$ of horizontal lines and $U$ of vertical lines such that each vertical line is cut by exactly two horizontal lines, and a positive integer $S \leq |H|$.
*Question:* Is there a set of horizontal lines, $H' \subseteq H$, such that every vertical line in $U$ is cut at least once and $|H'| \leq S$?

Second, the stick diagram is represented as a collection of adjacent rectangles and horizontal rays cutting all of the adjacencies in the collection of rectangles. These rays will be of two types "essential rays" which are the only rays to cut a particular adjacency and "choice rays" where a number of rays (none of which are essential) cut some adjacency. Not all of the choice rays are necessary to guard all of the adjacencies in the collection of rectangles. If it is possible to determine in polynomial time which of the set of choice rays guard all of the adjacencies in the diagram then it is possible to solve ***planar vertex cover*** in polynomial time — finding the minimum set of choice rays is equivalent to finding the minimum vertex cover of the original graph.

Proving that ***ray guard*** is NP-Complete is accomplished by means of two theorems — 4.1 which shows that ***stick diagram*** is NP-Complete using a transformation from ***planar vertex cover*** and 4.2 which shows that ***ray guard*** is NP-Complete using a transformation from ***stick diagram***.

It is however easier to perform the transformation from a planar graph to a stick diagram for a somewhat more restricted form of planar graph — a biconnected planar graph has properties which can be used in the transformation. Therefore it is desirable to prove one other result — vertex cover for a biconnected planar graph is NP-complete. Once it has been shown that this result holds the transformation from ***biconnected planar vertex cover*** to ***stick diagram*** and hence to ***ray guard*** can be done more easily. This result is addressed in Lemma 4.1.

***biconnected planar vertex cover***
*Instance:* Biconnected planar graph $G = (V,E)$, positive integer $B \leq |V|$.
*Question:* Is there a vertex cover of size $B$ or less for $G$, i.e. a subset $V' \subseteq V$ with $|V'| \leq B$ such that for each edge $\{u,v\} \in E$ at least one of $u$ and $v$ belongs to $V'$?

**Lemma 4.1** ***biconnected planar vertex cover*** *is NP-Complete*

**Proof**
Clearly ***biconnected planar vertex cover*** is in NP.

Now transform ***planar vertex cover*** to ***biconnected planar vertex cover***. Given a planar graph $G(E,V)$, $G$ can be converted to a biconnected planar graph $G'$ using the algorithm given in Figure 1. This is accomplished by ap-

```
G' ← G
j ← 1
while there exists a cut vertex v_j in G'
whose removal creates two components
X_j and Y_j (and possibly others)
    Let x_j ∈ X_j, y_j ∈ Y_j be such that x_j, y_j and v_j
        lie on a face
    Add a triangle graph, T_j, to the face in G'
        containing x_j, y_j and v_j.
    Add the edges (a_j,x_j) and (b_j,y_j) to G'
    j ← j+1
```

Figure 1: Creating a biconnected planar graph

propriate addition of instantiations of "triangle graphs" $T_j$ where each $T_j$ is defined as the vertices $a_j$, $b_j$ and $c_j$ and the edges $(a_j,b_j)$, $(b_j,c_j)$ and $(c_j,a_j)$. Clearly $G'$ is a biconnected planar graph since

1. after each iteration of the algorithm the vertex $v_j$ is no longer a cut vertex with respect to $X_j$ and $Y_j$ and
2. no new cut vertex is ever added during an iteration of the algorithm.

Thus the algorithm terminates with $G'$ free of all cut vertices. In addition, the triangle graphs $T_j$ which are added during each iteration are added to the face containing the vertices to which they are connected thus maintaining the planarity of the graph. Refer to Figure 4 for an example of a triangle graph and to Figure 5 for an example of how triangle graphs can be added to a planar graph using the algorithm in Figure 1 in order to derive a biconnected planar graph.

To determine the vertex cover for $G'$, the original graph $G$ plus the new edges and vertices must be considered. The structure of the triangle graphs means that for each triangle graph, $T_j$, exactly two of $a_j$, $b_j$ and $c_j$ must be in the vertex cover of $G'$. If $k$ triangle graphs are added then it is trivial to show that ***planar vertex cover***, $G$, $K$, is true if and only if ***biconnected planar vertex cover*** is true for $G'$, $B = K + 2k$.

The transformation from $G$ to $G'$ can clearly be accomplished in polynomial time. A cut vertex (articulation point) can be found in polynomial time [6] and there are at most $O(n)$cuts ($n$ is the number of vertices in $G'$).

Therefore ***biconnected planar vertex cover*** is NP-Complete.

□

***biconnected planar vertex cover*** can now be used to show that ***ray guard*** is NP-Complete. The proof is accomplished by means of the following two theorems — 4.1 and 4.2.

**Theorem 4.1** ***stick diagram*** *is NP-Complete.*

**Proof**

Clearly ***stick diagram*** is in NP — given a set of horizontal lines $H'$ such that $|H'| \leq S$, it is possible to check in polynomial time that every vertical line in $U$ is cut at least once.

Now transform ***biconnected planar vertex cover*** to ***stick diagram***. If $G(V,E)$ is a biconnected planar graph then $G$ can be embedded in the plane ($G$ is planar) and for every two vertices in $G$ an elementary cycle can be found which contains these vertices [1, 15] ($G$ is biconnected). This implies that there are no vertices of degree 1 in $G$ (other than the case where $G$ is a trivial graph with 2 vertices and 1 edge) and thus that all the faces in $G$ are bounded by cycles [15].

Let $F_0$ be the exterior face of $G$ and $F_1, \ldots, F_n$ be the interior faces of the graph. The biconnected planar graph $G$ can be transformed to a stick diagram by the following process.

1. Choose $C_s$ as being the cycle bounding any face, $F_s$ of $G$, which is adjacent to the exterior face $F_0$ of $G$.
2. Choose any two vertices $x$ and $y$ joined by an edge $A$ which form part of $C_s$ and are adjacent to the exterior. Represent $x$ and $y$ by horizontal lines in the stick diagram that cut the vertical line representing edge $A$ (see Figure 6 (a)).
3. Consider the path from $x$ to $y$, $B$, formed by removing edge $A$ from $C_s$. For the moment, treat $B$ as if it were a simple edge i.e. insert its corresponding vertical line into the stick diagram. This then gives the horizontal lines $x$ and $y$ cutting the vertical lines $A$ and $B$. The stick diagram is then as shown in Figure 6 (b).
4. Break the path $B$ (which was treated as a virtual edge) into its component edges. Let the path $B$ be the sequence of vertices $x, v_0, v_1, \ldots, v_k, y$. On the path $B$ from $x$ to $y$ whenever a vertex $v_i$ is encountered a new horizontal line must be added to the stick diagram. A new vertical line must also be added for each edge encountered. This is done in the following way. Suppose $C$ is the edge joining $x$ to $v_0$ along the path $B$. The stick diagram is now altered to include $C$. This is shown in Figure 6(c).

   In this case, $B'$ represents the original path $B$ minus the edge $C$ which has been included in the stick diagram. A similar operation is applied for all the edges on the path $B$. Each vertex $v_i$ maps to a horizontal line in the stick diagram and the edge joining it to the previous vertex is a vertical line cut by the two horizontal lines $v_{i-1}$ and $v_i$. After all the vertices on the path $B$ have been visited, the stick diagram will have the form shown in Figure 6 (d). A stick diagram which represents the originally selected closed region $F_s$ of the original graph has now been created.
5. If $G$ only had one interior face then the transformation is complete, otherwise continue with the next step
6. Let $CF$ (the composite of all the faces considered so far) be $F_s$
7. While there are still faces in $G$ to consider, repeat the following

(a) Choose a face $F_m, 1 \leq m \leq n$ that is adjacent to $CF$. $F_m$ must share a path with $CF$. $C_m$, the cycle enclosing $F_m$ is thus made up of two sets of vertices — those that are on the shared path and have already been "visited" (included in the stick diagram) and those that have not yet been visited. The vertices in the latter set make up the path $D$.

(b) Treat path $D$ as a single (simple) edge and add it to the stick diagram by extending the horizontal lines representing the start vertex and the end vertex of the shared path to cut a new vertical line representing the path $D$.

See Figure 6 (e) for an example — in this figure, a new path between $v_0$ and $v_k$ is being added.

The path $D$ can then be broken up into its constituent edges in the same fashion as before.

(c) Grow $CF$ by combining it with $F_m$ and removing the shared path between the faces.

This completes the construction of the stick diagram from a biconnected planar graph. A complete example of this is shown in Figure 7. First the face represented by $x-y-z-w$ is converted into a stick diagram. Then the face represented by $w-z-p$ is added to the stick diagram and finally the face represented by $w-p-z-y-q$ is added. This gives the complete stick diagram for the original biconnected graph.

Thus it can be seen that if a vertex cover, $V'$, can be found for $G$ then a set of horizontal lines, $H'$, can be found for $H$ — each vertex in $G$ is a horizontal line in $H$ and each edge in $G$ is a vertical line in $U$. Conversely if a set of horizontal lines $H'$, such that $|H'| \leq S$, could be found to cut each vertical line in $U$, then a vertex cover, $V'$, for $G$ could be found.

The transformation from ***biconnected planar vertex cover*** to ***stick diagram*** can be accomplished in polynomial time. Each face in the graph $G$ is considered in turn and once only. As the face is considered each edge is added in turn to the stick diagram as a vertical line — this happens once per edge. Horizontal lines are either added to the stick diagram to represent vertices or the horizontal line representing a vertex is extended as necessary. Each vertex can only occur in as many faces as there are in the graph and each vertex in each cycle is only visited once per cycle. Thus the number of operations on vertices is limited by a polynomial expression.

Therefore ***stick diagram*** is NP-Complete.

□

Theorem 4.1 shows that a stick diagram can be constructed for any biconnected planar graph. It is now necessary to show that a stick diagram can be represented by a collection of adjacent rectangles. This must be done in a manner that ensures consistency between a minimal selection of rays and a minimal selection of horizontal lines in the stick diagram. This is considered in Theorem 4.2 below. This theorem uses a construction from a stick diagram to produce a collection of adjacent rectangles in which the adjacencies between rectangles are cut by essential rays and choice rays where the choice rays are directly related to the horizontal lines in the stick diagram. Not all of the choice rays are necessary and Theorem 4.2 also shows that the problem of choosing the minimum number of such rays is NP-Complete.

**Theorem 4.2** ***ray guard*** *is NP-Complete*

**Proof**

Clearly ***ray guard*** is in NP. Given a set of rays it is possible to check in polynomial time that each adjacency has been cut by at least one ray.

Now transform ***stick diagram*** to ***ray guard***.

The transformation from a stick diagram to a collection of adjacent rectangles is done using the canonical choice unit shown in Figure 8. In this canonical choice unit, *ccu*, the essential rays originating in the four small rectangles and ending in the four darker shaded rectangles are enough to guard all of the adjacencies in the ccu except those between the middle rectangle and the two tall rectangles bordering it. These adjacencies can be cut by two possible maximal rays, only one of which is necessary. Scaling of the canonical choice unit does not change the fact that it can/does produce choice rays.

This transformation proceeds by replacing each vertical line in the stick diagram by a ccu of an appropriate size. The horizontal lines that cut through the vertical line are represented by the choice rays of the ccu. It is necessary to show that these canonical choice units can be joined together in a fashion which maintains the relation between the horizontal lines in the stick diagram and the choice rays in the configuration of adjacent rectangles. There are four ways in which horizontal lines could cut through successive vertical lines or in which choice rays could cut through successive adjacencies (actually there are only two ways, each with a vertical reflection). These are

1. the horizontal line could be the upper (lower) line through one vertical line and the upper (lower) line through the next vertical line (the upper (lower) choice ray of one ccu is the same as the upper (lower) choice ray of the next ccu),

2. the horizontal line could be the upper (lower) line through one vertical line and the lower (upper) line through the next vertical line (the upper (lower) choice ray of one ccu is the same as the lower (upper) choice ray of the next ccu).

In each of these cases it is possible to connect two ccu's in such a fashion that the relation between line in the stick diagram and choice rays is preserved. This is accomplished by making use of the darker shaded "connector" rectangles of each ccu (see Figure 8) and where appropriate making use of "connecting" rectangles. Figure 9 shows the construction for case 1 and its reflection. Figure 10 shows the construction for case 2 and its reflection.

If all of the vertical lines in the stick diagram are replaced by ccu's, connecting rectangles are added if appropriate

and the appropriate changes are made to the connector rectangles then the choice in the original stick diagram can be maintained. An example of converting a stick diagram to a collection of adjacent rectangles is shown in Figure 11.

The transformation from ***stick diagram*** to ***ray guard*** is thus accomplished by inserting an appropriately sized ccu for each vertical line and then joining these up by using the appropriate connecting rectangles working from the leftmost to the rightmost ccu, at each stage connecting the current ccu to those that have already been visited.

It is now necessary to show that there is a solution to ***stick diagram*** if and only if there is a solution to ***ray guard***. The construction of the collection of adjacent rectangles from the stick diagram changes the horizontal lines in the stick diagram to choice rays in the collection of rectangles. It also introduces 4 essential rays for every ccu added, these essential rays must be in the final solution to ***ray guard***. Suppose there is a solution for ***stick diagram***, i.e. there exists a set of lines $H'$ such that $|H'| \leq S$, then there must be a solution $P$ to ***ray guard*** with $|P| = |H'| + 4|U|$. This is because the essential rays must be in $P$ and the choice rays which correspond to the horizontal lines in $H'$ must also be in $P$. Conversely if there is a solution $P$ to ***ray guard*** then there must be a solution $H' = P - \{e \mid e$ is an essential ray in $P\}$ to ***stick diagram***.

This transformation can clearly be done in polynomial time — each vertical line is visited twice, once when it is replaced by a ccu and a second time when it is connected to the ccu(s) to its right in the stick diagram. If the stick diagram can be drawn then a configuration of ccu's can be drawn by scaling the ccu's to be the same size as the vertical lines that they represent. The ccu's (and their connecting rectangles) can thus be drawn as a non overlapping collection of adjacent rectangles — an instance of ***ray guard***.

***ray guard*** is thus NP-Complete.

□

This section shows that the general case of guarding all the adjacencies of a collection of adjacent rectangles with the minimum number of maximal-length rays (ray guarding) is NP-Complete. The next section of this article (Section 5) discusses special cases in which an exact solution can be found in polynomial time.

# 5 Special cases which can be solved in polynomial time

The orthogonal ray guarding problem is NP-Complete in general but there are some cases for which polynomial time algorithms can be obtained. In this section of the article some of these special cases are discussed.

Suppose that the union of the adjacent rectangles is itself a rectangle as in Figure 3 of Section 3 and Figure 12, then the ray guarding problem can be solved in polynomial time. This can be shown as follows. First, project all the vertical adjacencies onto a vertical line segment $L$ to obtain a set of intervals on $L$ (see Figure 13). Then the problem of finding the minimum number of horizontal lines that intersect the vertical adjacencies (the ray guarding problem) is equivalent to that of finding the minimum number of points on $L$ needed such that each interval contains at least one point. This is the problem of finding the minimum cover of an interval graph which can be solved in linear time [13]. The mapping from ray guarding to vertex cover for interval graphs is also possible for other configurations of adjacent rectangles provided that any vertical adjacencies which produce overlapping intervals when projected onto $L$ can be cut by a horizontal line which does not leave the union of the rectangles. For example for the configuration of rectangles in Figure 14 the mapping would produce a correct answer but in Figure 15 it would not.

In addition, Watts and Sanders [26] have shown that a chain of rectangles can be guarded in linear time if the input is given as a chain of adjacent rectangles.

# 6 Further Research

As extensions to the work done here there are a number of problems to be considered.

- Determining other cases of the orthogonal ray guard problem which can be solved in polynomial time.
- Addressing the related problem where each adjacency is only allowed to be cut by a single ray.
- Allowing the adjacencies to be guarded by non-orthogonal rays.
- Attempting to solve the more general problem of adjacent convex polygons.

# 7 Conclusion

This article addresses the problem of finding the fewest longest orthogonal line segments (maximal rays) that pass through all of the shared adjacencies between adjacent rectangles. The problem is made NP-Complete by the fact that various instances of choice can arise. The NP-Completeness proof is based on a transformation from planar vertex cover.

# Acknowledgements

The authors would like to thank the referees for their extremely helpful suggestions which were used in improving this article.

# References

[1] C Berge. *The Theory of Graphs And Its Applications*. Methuen & Co, London, 1962. Translated by A. Doig.

[2] I Bjorling-Sachs. 'A tight bound for edge guards in rectilinear monotone polygons'. Technical Report 93-12, Rutgers University, (1993).

[3] I Bjorling-Sachs and D L Souvaine. 'A tight bound for guarding general polygons with holes'. Technical Report LCSR-TR-165, Laboratory for Computer Science Research, Hill Centre for the Mathematical Sciences, Busch Campus, Rutgers University, New Brunswick, New Jersey, (1991).

[4] I Bjorling-Sachs and D L Souvaine. 'A tight bound for edge guards in monotone polygons'. Technical Report 92-52, Rutgers University, (1992).

[5] I Bjorling-Sachs and D L Souvaine. 'An efficient algorithm for guard placement in polygons with holes'. *Discrete & Computational Geometry*, **13**:77–109, (January 1995).

[6] G Brassard and P Bratley. *Fundamentals of Algorithmics*. Prentice Hall, Englewood Cliffs, New Jersey, 1996.

[7] J Culberson and R A Reckhow. 'Othogonally convex coverings of orthogonal polygons without holes'. *Journal of Computer and System Sciences*, **39**(2):166–204, (1989).

[8] J Culberson and R A Reckhow. 'Covering polygons is hard'. *Journal of Algorithms*, **17**:2–44, (1994).

[9] J Czyzowicz, E Rivera-Campo, N Santoro, J Urrutia, and J Zaks. 'Guarding rectangular art galleries'. *Discrete Mathematics*, **50**:115–120, (1995).

[10] D S Franzblau and D J Kleitman. 'An algorithm for covering polygons with rectangles'. *Information and Control,*, **63**:164–189, (1984).

[11] M R Garey and D S Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.

[12] L Gewali and S Ntafos. 'Covering grids and orthogonal polygons with periscope guards'. *Computational Geometry: Theory and Applications*, **2**:309–334, (1993).

[13] M C Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980.

[14] E Györi, F Hoffman, K Kriegel, and T Shermer. 'Generalized guarding and partitioning for rectilinear polygons'. *Computational Geometry: Theory and Applications*, **6**:21–44, (1996).

[15] F Harary. *Graph Theory*. Addison-Wesley, Reading, Massachusetts, 1969.

[16] B Hillier, J Hanson, J Peponis, J Hudson, and R Burdett. 'Space syntax, a different urban perspective'. *Architecture Journal*, **30**:47–63, (1983).

[17] D T Lee and A K Lin. 'Computational complexity of art gallery problems'. *IEEE Transactions on Information Theory*, (1986).

[18] D Lichtenstein. 'Planar formulae and their uses'. *SIAM Journal of Computing*, **11**(2):329–393, (May 1982).

[19] J O'Rourke. *Art Gallery Theorems and Algorithms*. Number 3 in The International Series of Monographs on Computer Science. Oxford University Press, New York, 1987.

[20] J O'Rourke. 'Visibility'. In J E Goodman and J O'Rourke, eds., *Handbook of Discrete and Computational Geometry*, pp. 467–479. CRC Press, Boca Raton, (1997).

[21] F P Preparata and M I Shamos. *Computational Geometry: An Introduction*. Texts and Monographs in Computer Science. Springer-Verlag, New York, 1985.

[22] T Shermer. 'Recent results in art galleries'. *Proceedings of the IEEE*, **80**(9):1384–1399, (September 1992).

[23] T C Shermer. 'Several short results in the combinatorics of visibility'. Technical Report 91-2, School of Computing Science, Simon Fraser University, (May 1991).

[24] S Suri. 'Polygons'. In J E Goodman and J O'Rourke, eds., *Handbook of Discrete and Computational Geometry*, pp. 429–444. CRC Press, Boca Raton, (1997).

[25] J Urrutia. 'Art gallery and illumination problems'. In J R Sack and J Urrutia, eds., *Handbook on Computational Geometry*. Elsevier Science. To appear.

[26] D C Watts and I D Sanders. 'Ray guarding a chain of adjacent rectangles'. Technical Report 97-02, University of the Witwatersrand, (1997).

[27] S Y Wu and S Sahni. 'Covering rectilinear polygons by rectangles'. *IEEE Transactions on Computer-Aided Design*, **9**(4):377–388, (1990).

Received: 2/10/97, Accepted: 8/6/98

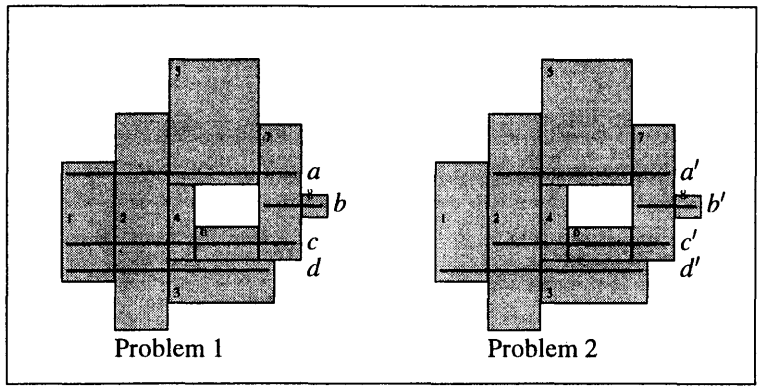


Figure 2: A simple configuration showing the two different problems

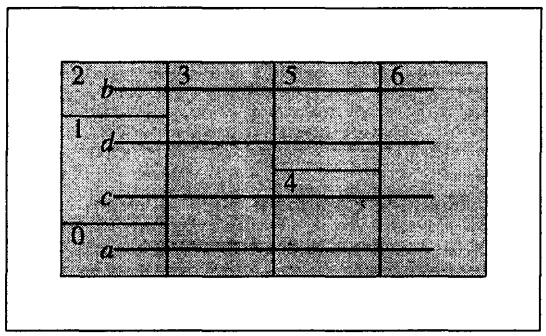


Figure 3: A configuration where the solution is not unique

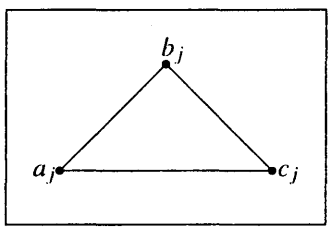


Figure 4: An example of a "triangle graph", $T_j$

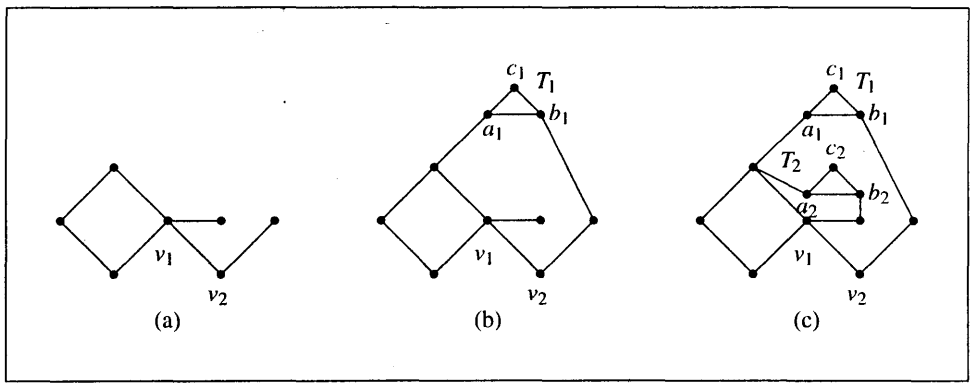


Figure 5: An example of adding triangle graphs to a graph to make it biconnected [(a) The orginal graph, $v_1$ and $v_2$ are cut vertices. (b) Graph with $T_1$ added, $v_1$ is still a cut vertex. (c) Final biconnected graph]

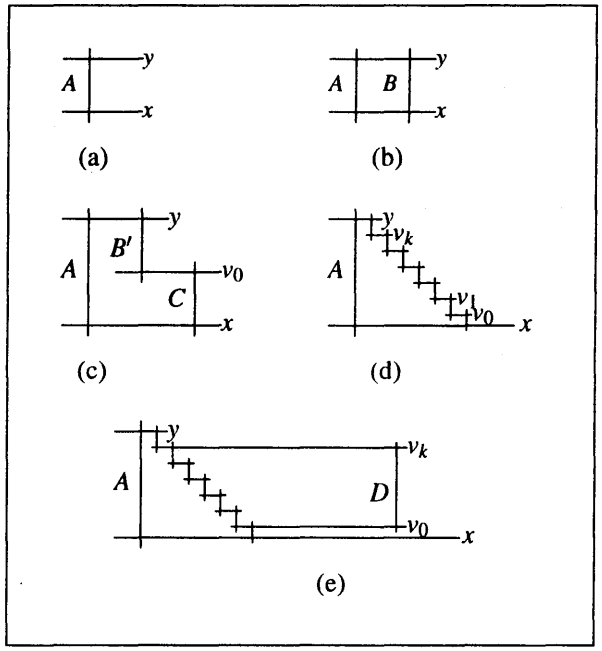


Figure 6: Creating a 'stick' diagram

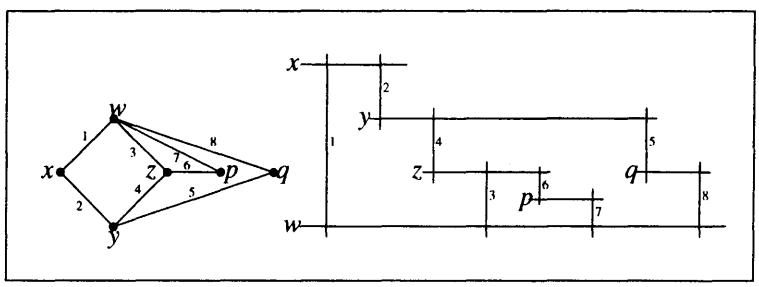


Figure 7: An example of the transformation of a biconnected planar graph to a stick diagram

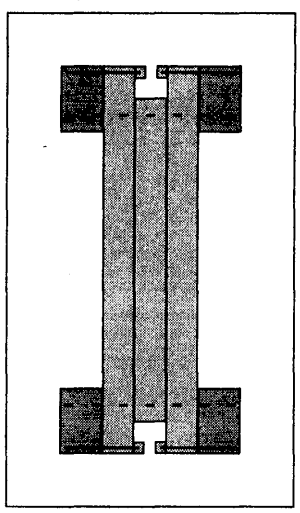

Figure 8: The Canonical Unit which produces two choice rays (shown as dashed lines)

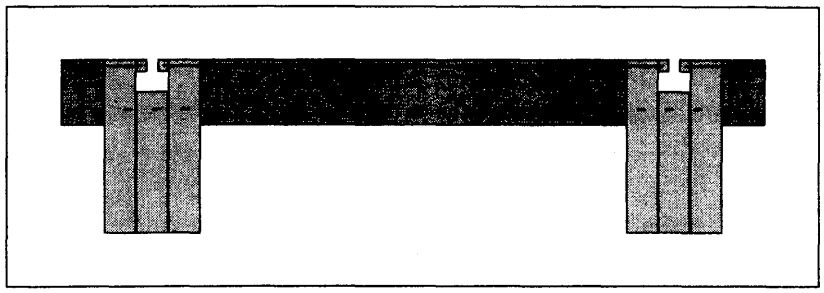

Figure 9: Joining the upper choice rays of two choice units

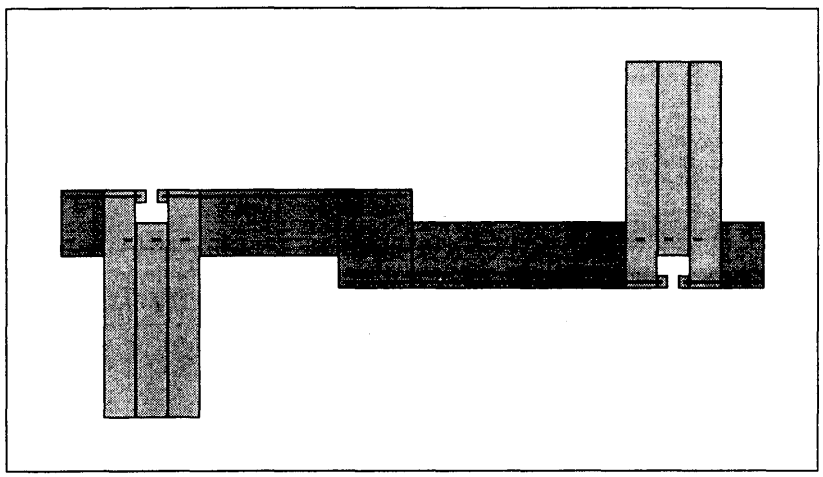

Figure 10: Joining the upper choice ray of one unit to the lower choice ray of the next unit

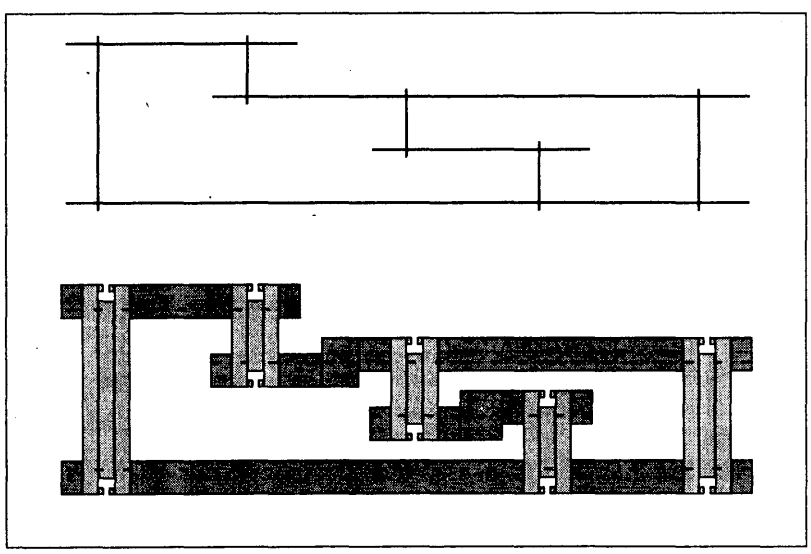

Figure 11: An example of converting a stick diagram to a collection of adjacent rectangles

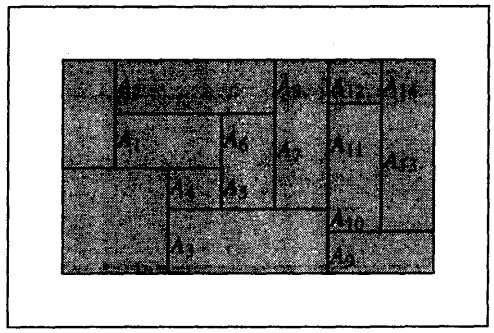

Figure 12: A simple configuration of rectangles with a rectangular union

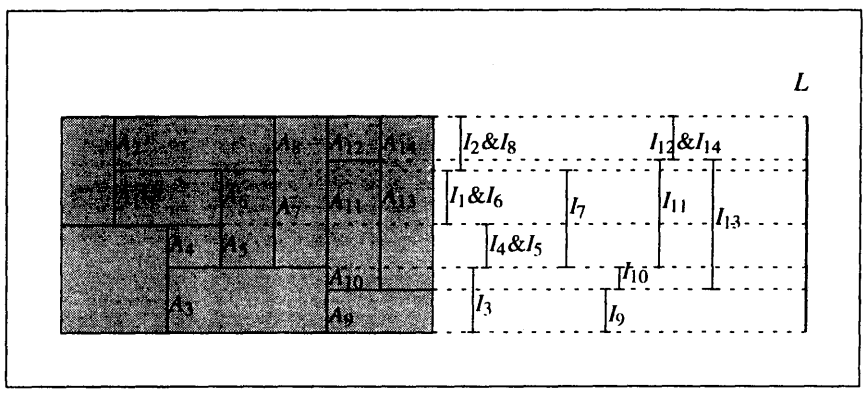


Figure 13: Projecting Adjacencies onto Intervals on the line *L*

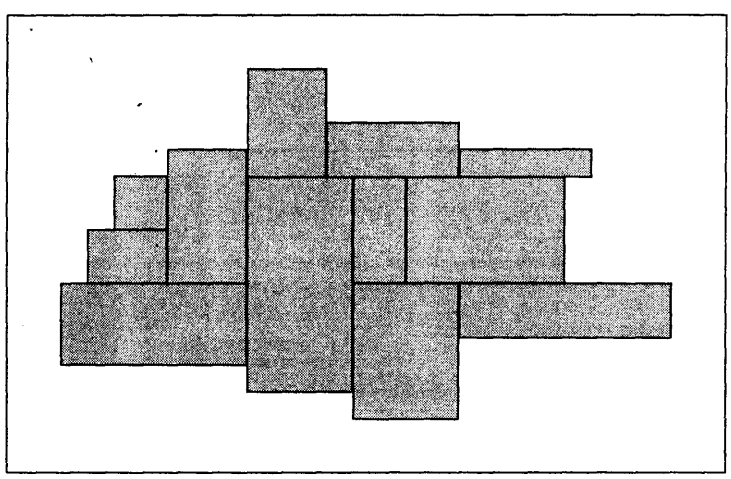

Figure 14: A simple configuration of rectangles which can be used in the production of an interval graph

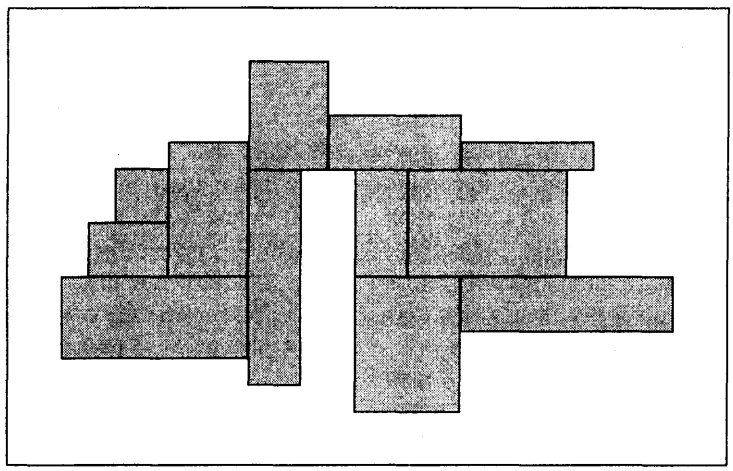

Figure 15: A simple configuration of rectangles which cannot be used in the production of an interval graph

# Notes for Contributors

The prime purpose of the journal is to publish original research papers in the fields of Computer Science and Information Systems, as well as shorter technical research notes. However, non-refereed review and exploratory articles of interest to the journal's readers will be considered for publication under sections marked as Communications of Viewpoints. While English is the preferred language of the journal, papers in Afrikaans will also be accepted. Typed manuscripts for review should be submitted in triplicate to the editor.

**Form of Manuscript**
Manuscripts for *review* should be prepared according to the following guidelines:

- Use wide margins and $1\frac{1}{2}$ or double spacing.
- The first page should include:
- the title (as brief as possible)
- the author's initials and surname
- the author's affiliation and address
- an abstract of less than 200 words
- an appropriate keyword list
- a list of relevant Computing Review Categories
- Tables and figures should be numbered and titled.
- References should be listed at the end of the text in alphabetic order of the (first) author's surname, and should be cited in the text according to the Harvard. References should also be according to the Harvard method.

Manuscripts accepted for publication should comply with guidelines as set out on the SACJ web page,

`http://www.cs.up.ac.za/sacj`

which gives a number of examples.

SACJ is produced using the LaTeX document preparation system, in particular LaTeX $2_\varepsilon$. Previous versions were produced using a style file for a much older version of LaTeX, which is no longer supported. Please see the web site for further information on how to produce manuscripts which have been accepted for publication.

Authors of accepted publications will be required to sign a copyright transfer form.

**Charges**
Charges per final page will be levied on papers accepted for publication. They will be scaled to reflect typesetting, reproduction and other costs. Currently, the minimum rate is R30.00 per final page for contributions which require no further attention. The maximum is R120.00, prices inclusive of VAT.

These charges may be waived upon request of the author and the discretion of the editor.

**Proofs**
Proofs of accepted papers may be sent to the author to ensure that typesetting is correct, and not for addition of new material or major amendments to the text. Corrected proofs should be returned to the production editor within three days.

**Letters and Communications**
Letters to the editor are welcomed. They should be signed, and should be limited to about 500 words. Announcements and communications of interest to the readership will be considered for publication in a separate section of the journal. Communications may also reflect minor research contributions. However, such communications will not be refereed and will not be deemed as fully-fledged publications for state subsidy purposes.

**Book Reviews**
Contributions in this regard will be welcomed. Views and opinions expressed in such reviews should, however, be regarded as those of the reviewer alone.

**Advertisement**
Placement of advertisements at R1000.00 per full page per issue and R500.00 per half page per issue will be considered. These charges exclude specialised production costs, which will be borne by the advertiser. Enquiries should be directed to the editor.

**South African Computer Journal**

Number 23, July 1999
ISSN 1015-7999

**Suid-Afrikaanse Rekenaar-tydskrif**

Nommer 23, Julie 1999
ISSN 1015-7999

# Contents