Searching a Solid Pseudo 3-Sided Orthoconvex Grid

Antonios Symvonis[1]* and Spyros Tragoudas[2]

[1] Basser Department of Computer Science, University of Sydney, N.S.W. 2006, Australia, symvonis@cs.su.oz.au
[2] Department of Computer Science, Southern Illinois University, Faner Hall, Carbondale, IL 62901, USA, spyros@buto.c-cs.siu.edu

Abstract. In this paper we examine the edge searching problem on pseudo 3-sided solid orthoconvex grids. We obtain a closed formula that expresses the minimum number of searchers required to search a pseudo 3-sided solid orthoconvex grid. From that formula and a rather straight forward algorithm we show that the problem is in P. We obtain a parallel version of that algorithm that places the problem in NC. For the case of sequential algorithms, we derive an optimal algorithm that solves the problem in $O(m)$ time where m is the number of points necessary to describe the orthoconvex grid. Another important feature of our method is that it also suggests an optimal searching strategy that consists of $O(n)$ steps, where n is the number of nodes of the grid.

1 Introduction

The *edge-searching* problem was introduced by Parson in [7]. An undirected graph was given and the objective was to clean its contaminated edges (or, in a different statement of the problem, to capture a fugitive hidden in them). Three kinds of actions were allowed in this cleaning operation:

1. *place(node):* This action places a searcher at the node specified as parameter of the action.
2. *pick(node):* This action picks up a searcher from the node specified as parameter of the action.
3. *move(origin, destination):* This action moves a searcher along the edge that connects the *origin* and the *destination* nodes. For the action to be legal, the two nodes must be connected by an edge and a searcher must be initially located at the *origin* node.

The *search number* of a graph G, denoted by $es(G)$, was defined in [6] as the minimum number of searchers that are required in order to clean the graph (or capture the fugitive that is hidden in its edges). In that paper it was proven that the decision problem *" Given a graph G and an integer k, can G be cleared with k searchers?"* is NP-Hard. The authors also pointed out that the problem

* Supported by the University of Sydney Research Grant Scheme

would belong in the class NP if it was true that recontamination cannot help in searching a graph. We say that a clean edge is *recontaminated* if it becomes adjacent with a contaminated edge and no searcher is placed at their common node. Recontamination can start when a searcher that is positioned at a node adjacent to a clean edge and at least one contaminated edge, leaves the node (either by a *pick* or *move* action) and allows the clean edge to be contaminated again. We assume that recontamination propagates at an infinite speed, i.e., if recontamination occurs as a result of an action t of the searching, then, before action $t+1$, all edges that can become contaminated again will do so. LaPaugh [4] proved that recontamination does not help to search a graph and thus the edge searching problem was included in the class NP. Besides its theoretical importance, this result is useful in the sense that allows us to assume that there exists a strategy that searches the graph using the minimum number of searchers and never allows recontamination. A consequence is that the graph can be searched in a finite number of actions. After LePaugh's work a great deal of effort was devoted to the searching problem. Most of the results related the searching problem with other combinatorial optimization problems such as *pebbling* [2], *cutwidth* [5] and *graph separators* [1].

In this paper, we concentrate on the searching problem on a special kind of graphs, namely, the pseudo 3-sided solid orthoconvex grids (Section 3.1). We show how to determine the search number of such graphs in optimal time. We do that by defining a modified version of the edge searching problem which we call *modified edge searching* (Section 2). Then, for the modified edge searching problem, by proving that there are searching strategies that possess several properties regarding the way the grid is searched, we are able to obtain a closed formula that expresses the minimum number of searchers required to search a pseudo 3-sided solid orthoconvex grid (Section 3.2). From that formula we derive an algorithm that computes $es()$ in polynomial time (Section 4).

We can also obtain a parallel version of that algorithm that places the problem in NC. For the case of sequential algorithms, we derive an optimal algorithm that solves the problem in $O(m)$ time where m is the number of points necessary to describe the orthoconvex grid. Another important feature of our method is that it also suggests an optimal searching strategy that consists of $O(n)$ steps, where n is the number of nodes of the grid.

Previously, we were able to determine the searching number in optimal time only for the class of trees [6]. We were also able to solve the decision problem *"Given a graph G of n nodes can we search G by using a constant number of k searchers ? "* in $O(n^2)$ [3]. We improve this result for the case of pseudo 3-sided solid orthoconvex grids. Some work has already been done for search problem in rectangular grids [9]. However, in that work the searchers are more powerful than the ones we use (i.e., different actions are assumed) and contamination does not propagate in an infinite speed. Furthermore, the rectangular grid that was assumed as the underlying graph structure belongs in the class of the pseudo 3-sided solid orthoconvex grids.

Because of space limitations, we omit all proofs of lemmata and theorems. Someone interested in them can refer to [8].

2 A New Version of the Searching Problem

In this section, we define a new version of the searching problem which we call *modified edge searching*. The difference between the two searching problems are in the possible actions that can take place during the search.

Definition 1. We say that we have a *modified edge searching problem on a graph* G if we are allowed to search the graph using all 3 actions of the original edge searching problem as well as the additional 4^{th} one:

4. *clean*($node1, node2$)*:* This action cleans edge ($node1, node2$) or the path ($node_1, node_i$), ($node_i, node_2$) where $node_i$ is of degree 2. For the action to be legal searchers must be placed on both $node1$ and $node2$.

Kirousis and Papadimitriou [2] defined a similar searching problem which they called *node searching*. In their version of the game only *place, pick* and *clean* actions were allowed and the clean action could clean only one edge.

Definition 2. A *searching strategy* $S(G)$ is a sequence of actions $< a_1; \ldots; a_m >$ such that when applied on a graph G which has all of its edges contaminated has the effect to clean the edges of G. Action $a_i, 1 \leq i \leq m$, is any action allowed in the searching problem. A searching strategy is said to be *optimum* when there is no other strategy that uses a smaller number of searchers and also searches the graph.

Two searching strategies are *equivalent* if they both search the same graph with the same number of searchers.

We say that a node is *clean* if all of its adjacent edges are clean. A node is *dirty* if all of its adjacent edges are contaminated. If it is neither clean nor dirty we say that it is *partially clean*.

A *move*($origin, destination$) action is *useless* if it neither cleans an edge nor results into contamination. A useless *move* action occurs when the searcher moves from a clean node, or, from a partially clean node to a clean one and is not causing recontamination, or, from a dirty node that has no other searcher.

We say that a searcher is *useful for action* a_t (or simply that it is useful at time t) if its removal will cause recontamination, or it will prevent action a_t of happening. Otherwise, we say that the searcher is *useless*.

We can prove the following lemmata:

Lemma 3. *There is an optimum searching strategy for the (modified) edge searching problem on graph G that contains only useful move actions.* □

Lemma 4. *There is an optimum searching strategy for the modified edge searching problem on graph G that has the property that no useless searcher is on G immediately before any place, move, or clean action.* □

Corollary 5. *There is an optimum searching strategy for the original edge searching problem on graph G that has the property that no useless searcher is on G immediately before any place, or move action.* □

Lemma 6. *If there is a searching strategy that solves the modified edge searching problem on graph G using k searchers then there is a searching strategy that solves the edge searching problem on graph G using either k or $k+1$ searchers.* □

3 Searching strategies for Pseudo 3-Sided Solid Orthoconvex Grids

3.1 Definitions

Let $G^{\infty}(V_{\infty}, E_{\infty})$ be the infinite undirected graph whose node set V_{∞} consists of all points of the plane with integer coordinates and in which two vertices are connected by an edge in E_{∞} if and only if the Euclidean distance between them is equal to one. Let $G_i(V_i, E_i)$ be a finite node-induced subgraph of G.

A *grid graph* $D(V, E)$ is a subgraph of G_i where, $V = V_i$ and $E \subseteq E_i$. In the following discussion we will consider only graphs that are connected and all of their nodes have degree greater than 1. We say that a grid graph $D(V, E)$ is *solid* if it has no holes.

If we color black all unit squares in G^{∞} that are subrounded by the edges of a solid grid graph D, we will divide the plane into two regions, one black and one white. A node v that belongs into a solid grid graph and is adjacent to both the black and the white region is said to be a *boundary node*. The set of all boundary nodes of a solid grid D is said to constitute the boundary of D.

Assume a solid grid graph D and its corresponding black and white regions. D is said to be *orthoconvex* if and only if the intersection of any line parallel to any coordinate axis with the black region consists of at most one line segment.

A node v at the boundary of a solid grid graph is said to be a *convex boundary node* if it is of degree 2 and a *concave boundary node* if it is of degree 4. A node v at the boundary of a solid grid graph is said to be a *turning boundary node* if it is either a convex or a concave boundary node. Otherwise, it is called a *simple boundary node*.

From the above definitions it is obvious that a solid grid graph can be completely defined by the coordinates of its turning boundary nodes. In the rest of the paper, we assume that the grid under consideration is represented by its turning boundary nodes given in the order they appear if we traverse its boundary in the clockwise direction. An arbitrary node is selected to be the start of the traversal.

During our traverse of the boundary of a solid grid graph and assuming that we always look towards the next boundary node, we have to make several turns. So, the traversal of a solid grid can be represented by a word which has length equal to the number of turning boundary nodes over an alphabet of two letters namely, L for left and R for right. We call that word the *coding* of the solid

orthoconvex grid. Since we can start our traversal of the boundary of the grid from any turning point, it is useful to think of the coding as a circular word where the first and the last characters wrap around.

Since we can return to the point from which we started, it is obvious that we have 4 more R's than L's. Also observe that in an orthoconvex grid we never have two consecutive L's in its coding. By canceling each L and the R that follows it in the coding of a solid orthoconvex grid, we are left with 4 R's. These correspond to 4 convex boundary nodes. These points define the sides of the grid. In that sense, all orthoconvex grids are 4-sided.

However, we can relax that definition for the special case of the solid orthoconvex grids that contain the patterns RRRR or RRRRR. In these two cases, we can combine 2 sides together and thus, consider the orthoconvex to be composed by a *base*, a *rising region* that is immediately after the base in the clockwise direction, and a *falling region* that follows the rising region in the clockwise direction. For this reason, we call all the solid orthoconvex grids that fall into that category *pseudo 3-sided*. In the following we will refer to the boundary nodes that lie between any two sides (or pseudo sides) as *corners*. Figure 1 shows the two types of pseudo 3-sided solid orthoconvex grids along with their codings, corners and sides.

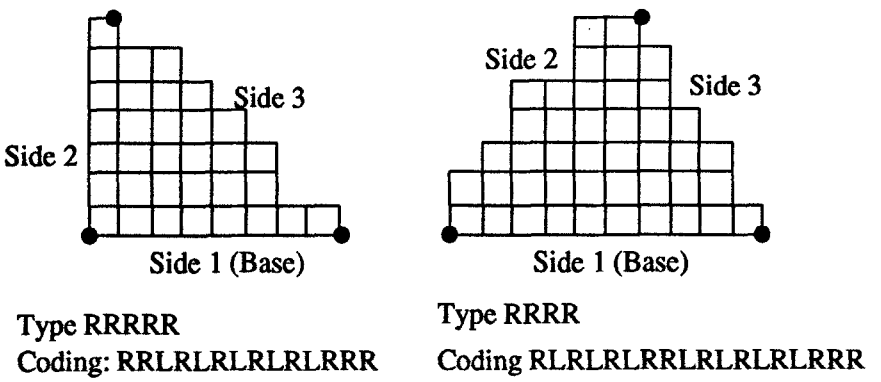

Fig. 1. The two types of pseudo 3-sided solid orthoconvex grids and their codings.

A *cord* is any path (possibly a closed one) internal to the grid that consists of grid edges as well as line segments that connect nodes that are of distance 2 and diagonally opposite of each other. If it is not a closed one it must have its endpoints on the boundary.

A *diagonal* is a cord that has its end-nodes on two different sides. We make the convention that a convex boundary node that separates two sides belongs in both of them and, in that sense, it is considered to be a diagonal as well. It is obvious that the nodes that belong into a nontrivial diagonal that touches each of its adjacent side at most once form a cut-set for the solid grid.

During the course of the searching there are regions of the grid that are *clean*

and others that are considered to be *contaminated* (or *dirty*).

Definition 7. Assume a searching strategy $S(G)$ on a graph G. Let D_c^t be the graph that is induced by removing all cleaned edges and all nodes that have no incident contaminated edges after t steps of $S(G)$. $Conn(D_c^t)$ is the number of connected components of D_c^t and it denotes also the number of contaminated regions at time t. We say that a graph can be searched in such a way that we have at most μ contaminated regions if and only if $\max_{t>0} Conn(D_c^t) \leq \mu$.

3.2 Searching strategies for Pseudo 3-Sided solid Orthoconvex Grids

In this section we show that there exists an optimum searching strategy for the modified edge searching problem on a pseudo 3-sided solid orthoconvex grid D in which during the course of the searching there exists only one contaminated region. Based on that, we compute $mes(D)$ and we show how from it to derive $es(D)$.

We can prove that[8]:

Lemma 8. *Assume a solid orthoconvex grid and a cord that has its end-nodes on the same side of the grid. We can search the region that is bounded from that side and the cord using the modified edge searching with a number of searchers equal to the cord points and in such a way that one searcher ends up at every cord node.* □

Corollary 9. *Assume a solid orthoconvex grid and a cord that has its end-nodes on the same side of the grid. We can search the region that is bounded from that side and the cord using the modified edge searching with a number of searchers equal to the cord points and in such a way that one searcher starts from each cord node.* □

Lemma 10. *Assume a pseudo 3-sided solid orthoconvex grid D and a diagonal. We can search the region of D that is bounded from the diagonal and the part of the boundary that contains exactly 1 corner using the modified edge searching with a number of searchers equal to the diagonal points and in such a way that one searcher ends up at every diagonal node.* □

Corollary 11. *Assume a pseudo 3-sided solid orthoconvex grid D and a diagonal. We can search the region of D that is bounded from the diagonal and the part of the boundary that contains exactly 1 corner using the modified edge searching with a number of searchers equal to the points of the diagonal and in such a way that one searcher starts from each point of the diagonal.* □

Lemma 12. *Any solid grid D that has b boundary nodes can be searched with b searchers in such a way that one searcher starts at every boundary node, or, one searcher ends at every boundary node.* □

Using the above lemmas, we can prove [8] the following theorem that will enable us to obtain a closed formula for $mes()$.

Theorem 13. *There is an optimal searching strategy for the modified edge searching problem on a pseudo 3-sided solid orthoconvex grid which has the property that during the search there is only one contaminated area.* □

Let $d(n_1, s)$ be the length of a shortest diagonal from the boundary node n_1 to side s. $d(n_1, s) = 0$ if n_1 is on side s. If node n_1 is a corner then let $s(n_1)$ denote the side which is opposite of it. Let e denote a boundary edge (n_1, n_2) or a path $(n_1, n_i), (n_i, n_2)$ where n_i is of degree 2. Let $S_a(e)$ to denote the side that is following the one that e lies on if we move from n_1 to n_2, and $S_b(e)$ to denote the side that is following that e lies on if we move from n_2 to n_1. In order for $S_a()$ and $S_b()$ to be well defined, e must not be a path of length 2 that contains a corner. In that case (e lies on two sides) we define $S_a(e)$ $(= S_b(e))$ to be the third side of the pseudo 3-sided solid orthoconvex grid.

Based on Theorem 13 we prove:

Theorem 14. *The minimum number of searchers that are required in order to solve a modified edge searching problem on a pseudo 3-sided solid orthoconvex grid D is given by:*

$$mes(D) = \min(corner\_distance, diagonal\_pair\_distance)$$

where, $corner\_distance = \min\{d(c, s(c))\}$ over any corner c (out of 3 possible) and $diagonal\_pair\_distance = \min\{d(n_1, S_b(e)) + d(n_2, S_a(e))\}$ over any boundary edge $e = (n_1, n_2)$ or any path $e =< (n_1, n_i), (n_i, n_2) >$ where n_i is of degree 2. □

Up to now, we have determined $mes(D)$, the minimum number of searchers that are required in order to solve the modified edge searching problem on a pseudo 3-sided solid orthoconvex grid. By Lemma 6 we know that at most $mes(D) + 1$ searchers can solve the original edge searching problem. So, we have a way to approximate $es(D)$ within 1 from the optimum. Unfortunately, as Figure 2 demonstrates, there are grids that can be searched optimally with the same number of searchers on both problems. So, in order to get an algorithm that computes $es(D)$ we have to identify these grids.

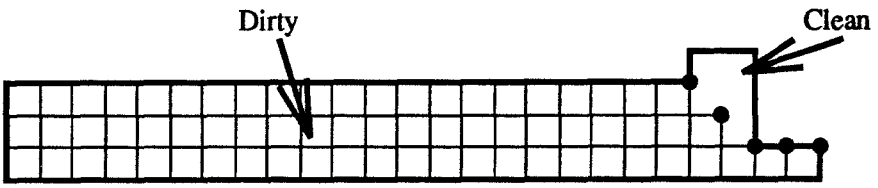

Fig. 2. A grid D for which $mes(D) = es(D) = 5$.

Lemma 15. *If there is a searching strategy that solves the edge searching problem on grid D that contains no path of the form $(n_1, n_i), (n_i, n_j), (n_j, n_2)$ where $degree(n_i) = degree(n_j) = 2$ using k searchers, $k > 2$, then there is a searching strategy that solves the modified edge searching problem on grid D using at most $k - 1$ searchers.* □

Lemma 16. *Assume a pseudo 3-sided solid orthoconvex grid D that contains a path of the form (n_1, n_i), $(n_i, n_j), (n_j, n_2)$ where $degree(n_i) = degree(n_j) = 2$ and n_j is a corner, and the diagonal from n_j to the opposite side has $mes(D)$ points. Then, $mes(D) = es(D)$ if and only if there are two points in the diagonal with the same X or Y-coordinate.* □

Theorem 17. *Assume that the minimum number of searchers that are required to solve the modified edge searching problem on a pseudo 3-sided solid orthoconvex grid D is $mes(D)$. Then, in the case where there is a diagonal from a corner c of D to the opposite side of length $mes(D)$ such that c is next to a degree 2 node and that diagonal has two points with the same X or Y-coordinate, $es(D) = mes(D)$. Otherwise, $es(D) = mes(D) + 1$.* □

4 Algorithms for determining es(D)

In this section we present algorithms to determine $es(D)$. The algorithms are based on Theorems 14 and 17. Through their proofs [8], these theorems also suggest an optimum searching strategy that consists of $O(n)$ actions where n is the number of nodes of the grid D.

It is customary to express the complexity of an algorithm that determines the minimum number of searchers that are required to search a graph as a function of n. However, for a grid we can define two new quantities: the number of boundary nodes b and the number of turning points m. Obviously m turning nodes can completely define the grid and thus it is desired to express the time complexity of the algorithm that determines $es(D)$ as a function of m. Observe that there are grids with $m = o(b)$ and $b = o(n)$.

A quantity that we have to compute is the distance from a boundary node v to some side of the grid. In the rest of the paper we assume that the base of the grid is parallel to the X axis. Informally, we describe how this can be done when the boundary node is on the rising side and we want to compute is distance to the falling side. All other cases are handled in a similar way. We move from node v diagonally up (on a line parallel with $l : x = y$) until we hit the boundary. If we hit the wanted side we are done. If not, we move to the right at the next concave turning point. We then move diagonally up, and so on. From the above discussion, it becomes obvious that we need to compute the distance from any concave turning point to any side.

The following lemma is important for deriving efficient algorithms:

Lemma 18. *Assume a pseudo 3-sided solid grid D that has its base parallel to the X axis. In order to determine $mes(D)$ as Theorem 14 indicates, it is*

sufficient to examine diagonals that start from i) turning nodes, ii) boundary nodes that are adjacent to turning nodes, and iii) the nodes that are at the intersection of the base and all lines that pass from concave turning nodes on the rising side and are parallel with $l : y = -x$.

It is trivial to compute the minimum diagonal between any boundary node and the base of the grid since one minimum diagonal will be parallel to the Y axis. The nontrivial part is to compute the the minimum diagonal from a concave turning node that lies on the the rising (falling) side to the falling (rising) side. *Algorithm_1* [8] supports the following lemma:

Lemma 19. *Given the side s of a pseudo 3-sided solid orthoconvex grid D that contains m turning nodes, we can compute the length of the minimum diagonals between all turning nodes x and s in $O(m^2)$ steps.* □

We now proceed to construct *Algorithm_2* that computes $es(D)$ based on $mes(D)$ as defined in Theorem 14.

From Lemma 18, we know that we can concentrate only at $O(m)$ nodes of the grid. Recall the definitions from Section 3.2 (following Theorem 13).

Algorithm_2

1. Compute the length of the minimum diagonals between all nodes v indicated in Lemma 18, and any side s.
2. *corner_distance* $= \min\{d(c, s(c))\}$ over any corner (out of the 3 possible) c.
3. *diagonal_pair_distance* $= \min\{d(n_1, S_a(e)) + d(n_2, S_b(e))\}$ over any boundary edge $e = (n_1, n_2)$ adjacent to a node specified in Lemma 18, or any path $e =< (n_1, n_i), (n_i, n_2) >$ where n_i is of degree 2 and thus, a convex turning node).
4. $mes(D) = \min($*corner_distance*, *diagonal_pair_distance*$)$.
5. Determine $es(D)$ from $mes(D)$ based on Theorem 17.

Lemma 20. *Given a pseudo 3-sided solid orthoconvex grid D, $es(D)$ can be determined in $O(m^2)$.* □

Algorithm_2 can be parallelized to yield a parallel algorithm that runs in polylog time using a polynomial number of processors. So we can state:

Theorem 21. *Given a pseudo 3-sided solid orthocnvex grid D, the problem of determining $es(D)$ is in NC.* □

We can modify *Algorithm_1* to compute the length of the minimum diagonals between all turning points v and any side s in $O(m)$ time by a more complicated method that is not easy (if possible) to be parallelized [8]. This improvement leads to an optimum algorithm and together with the fact that Theorem 14 suggests a searching strategy, our final result can be stated as:

Theorem 22. *Given a pseudo 3-sided solid orthocnvex grid D, an optimum edge searching strategy $S(D)$ that consists of $O(n)$ actions can be constructed. It uses $es(D)$ searchers where, $es(D)$ can be computed optimally in $O(m)$ time.*

5 Conclusion

We have shown how to compute the searching number for the class of the pseudo 3-sided solid orthoconvex grids. Our algorithms also suggest an optimal searching strategy. It is not clear how to search any orthoconvex grid. Especially, the property that allowed us to design our algorithms does not hold for every orthoconvex grid. We are able to create orthoconvex grids that cannot be searched by maintaining only one clean area during the searching. However, our conjecture is that there are optimum searching strategies that create at most two dirty areas. This can lead to algorithms for computing the searching number of any solid orthoconvex grid. Other interesting problems are i) how to search general solid grids(non orthoconvexes), and ii) how the existence of holes in the grid can effect the complexity of the problem.

References

1. J.A. Ellis, I.H. Sudbourough and J.S. Turner, "Graph Separation and Search Number".
2. L.M. Kirousis and C.H. Papadimitriou, "Searching and Pebbling", Theoretical Computer Science, 47, 1986, pp. 205-218.
3. M.R. Fellow and M.A. Langston, "Nonconstructive Tools for Proving Polynomial-Time Decidability", Journal of the Association for Computing Machinery, Vol. 35, No. 3, July 1988, pp. 727-739.
4. A.S. LaPaugh, "Recontamination Does Not Help to Search a Graph", Technical Report, Electrical engineering and Computer Science Department, Princeton University, 1983.
5. F. Makedon and I.H. Sudborough, "On Minimizing Width in Linear Layouts", Discrete Applied Mathematics, 23, 1989, pp. 243-265.
6. N. Megiddo,S.L. Hakimi, M.R. Garey, D.S. Johnson and C.H. Papadimitriou, "The complexity of Searching a Graph", Proceedings of the 22nd IEEE Foundations of Computer Science Symposium, 1981, pp. 376-385.
7. T.D. Parsons, "Pursuit Evasion in a Graph", in Theory and Applications of Graphs, Y. Alavi and D.R. Lick, eds., Springer Verlag, 1976, pp.426-441.
8. A. Symvonis and S. Tragoudas,"Searching a Solid Pseudo 3-Sided Orthoconvex Grid", Technical Report 438, Basser Department of Computer Science, University of Sydney, May 1992.
9. K. Sugihara and I. Suzuki, "Optimal Algorithms for a Pursuit-Evasion Problem in Grids", SIAM Journal of Discrete Mathematics, Vol. 2, No. 1, February 1989, pp. 126-143.