Fast Heuristics for Covering 1.5D Terrain

Laxmi Gewali and Jiwan Khatiwada

Abstract

We review important algorithmic results for the coverage of 1.5D terrain by point guards. Finding the minimum number of point guards for covering 1.5D terrain is known to be NP-hard. We propose an approximation algorithm for covering 1.5D terrain by a few number of point guards. The algorithm which we call Greedy Ranking Algorithm is based on ranking vertices in term of number of visible edges from them. We also present an improvement of the Greedy Ranking Algorithm by making use of visibility graph of the input terrain.

Keywords

Terrain visibility · Tower placement algorithm · Terrain illumination

75.1 Introduction

Problems related to visibility on terrain surface have numerous applications such as (1) geographic data frameworks, (2) route management for aerial vehicles, (3) transportation systems, (4) crisis reaction arranging, and (5) remote communications system (Wired and Wireless).

Terrain visibility problems can be viewed as a restricted instance of the well-known Art Gallery Problem [1] in computational geometry. In the art gallery problem, the domain is a simple polygon and it is required to find the set S of a minimum number of point guards inside the polygon so that any point inside it is visible from some point in S. It is remarked that two points p_i and p_j inside a polygon are visible to each other if the line segment having p_i and p_j as endpoints does not intersect with the exterior of the polygon. The standard art gallery problem is known as NP-hard [1]. This intractability result has motivated many researchers to look for approximation algorithms for art gallery problem [1]. Some variation of the standard art gallery problem has been considered. Such variations include an alternative notion of visibility and having input polygon restricted to monotone polygons and orthogonal polygons. In visibility variations, the notion of staircase visibility [2, 3]. In the staircase visibility model, two points p_i and p_j inside the polygon are visible if there is a staircase path connecting p_i and p_j that lies completely inside the polygon. It is noted that in a staircase path the edges are parallel to x-axis and y-axis and the path itself is monotone.

In this paper, we use the standard notion of visibility and restrict the polygonal domain as a monotone polygon in which one chain is a monotone chain and the other chain is a line segment. A monotone polygon with one chain as line segment is precisely a 1.5D terrain. The standard terrain is a 2.5D structure which means that a terrain is a structure which is between two dimensions and three dimensions. This view can be further elaborated in term of the cross-section of terrain with a horizontal plane. If we consider the cross-section of a terrain with a horizontal plane then the cross-section area become progressively smaller as the height of the horizontal plane increases.

The paper is organized as follows. In Sect. 75.2, we review important existing algorithms dealing with the visibility property of simple polygon and 1.5D terrains. In particular, we examine existing algorithmic results for placing guards to cover 1.5D terrains. We also review intractability results and approximation algorithms for placing point guards in a monotone polygon and 1.5D terrain. In Sect. 75.3, we present the main algorithmic result. We design, describe and sketch an approximation algorithm for finding a reduced number of point guards to cover (or illuminate) a 1.5D terrain. The

L. Gewali (✉) · J. Khatiwada
Department of Computer Science, University of Nevada, Las Vegas, NV, USA
e-mail: laxmi.gewali@unlv.edu

S. Latifi (ed.), *17th International Conference on Information Technology–New Generations (ITNG 2020)*, Advances in Intelligent Systems and Computing 1134,
https://doi.org/10.1007/978-3-030-43020-7_75

algorithm which we call "Greedy Ranking" is based on the ranking of vertices on visibility measures. The nodes are then processed in a greedy manner by placing the first point guard at the node with the largest visibility and other guards are progressively placed by re-ranking the uncovered nodes. The time complexity of the algorithm is $0(|E|log|V|)$ where $|E|$ is the number of edges and $|V|$ is the number of vertices in visibility graph induced by the terrain. Finally, in Sect. 75.4, we discuss (1) possible extension of the proposed algorithms and (2) interesting variations of the terrain illumination problem for future research.

75.2 Preliminaries

Problems dealing with visibility in the presence of polygons has been investigated by several researchers since last 40 years [4, 5]. In defining the notion of the visibility, the boundary of a polygon is considered as an opaque object. Two points inside the polygon are visible if the line segment connecting them do not intersect with the boundary. One of the widely investigated visibility problems on the simple polygon is to illuminate the entire interior of the polygon by placing a minimum number of point guards inside the polygon boundary. This is often known as the *Art Gallery* problem. Interested readers can find such problem in [4, 5].

The problem of placing the minimum number of guards inside a simple polygon is known to be intractable [6]. This problem remains NP-hard even for some restricted classes of polygons. One of the widely studied restricted class of simple polygons are monotone polygons. A simple polygon is called monotone if its boundary can be partitioned into two chains, each of which are monotone with respect to a given direction. Monotone polygons are used to model two dimensional terrain. Finding the minimum number of point guards to cover terrain has applications in telecommunication tower placement and geographic information system. An instance of the placement of point guards on a 1.5D terrain is shown in Fig. 75.1. In this problem instance, five point guards are needed. The point guards are drawn as small circles. Readers can easily verify that the terrain cannot be illuminated (or covered) with less than five point guards. The problem of finding the minimum number of point guards in terrain was a long standing open problem, which was settled by James King and Erik Krohn in 2009. They proved [7] this problem to be NP-hard.

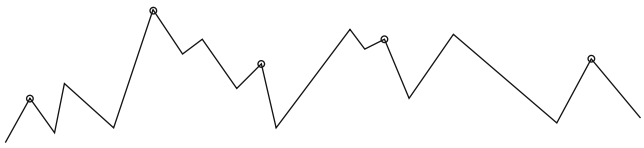


Fig. 75.1 An instance of terrain illumination

They reduced an instance of PLANAR 3-SAT problem to an instance of minimum guard placement problem in the monotone polygon. PLANAR 3-SAT problem is a restricted version of the standard 3-SAT problem [8]. In a planar 3-SAT, the graph implied by the satisfiability expression has to be a planar graph. Both standard 3-SAT and PLANAR 3-SAT problem are known to be NP-Hard [8]. A related problem on tower placement is reported in [9].

75.2.1 One-Sided Versus Two-Sided Guarding

The standard terrain or terrain guarding problem is the one-sided guarding problem. In the definition of one-sided guarding problem, a point p_i in the domain is said to be guarded if p_i is visible from any guard.

Very recently [10] the notion of two-sided guarding problem has been introduced in the context of guarding 1.5D terrain. In this definition, a point p_i in the terrain is said to be two-sided guarded if p_i is visible to at least one guard in the left and at least one guard in the right. The distinction of one-sided guarding and two sided guarding is shown in Fig. 75.2

Distinguishing one-sided and two-sided guarding

An examination of the terrain in Fig. 75.2 shows that it needs 5 guards (X) to cover it under two-sided notion of visibility. This terrain can be guarded by two guards (shown by o) under normal (one-sided) notion of visibility. While finding the minimum number of guards to cover 1.5D terrain is NP-Hard [7], the problem can be solved in linear time [10] under the notion of two-sided visibility.

75.2.2 Approximation Algorithms

The intractability of the art gallery problem has motivated many authors to develop approximation algorithms. One of the first such algorithm was proposed by S.K Ghosh [11]. This paper contains approximation algorithms for both simple polygons and polygon with holes. It is established in [11] that a simple polygon with n vertices can be guarded with numbers of guards m such that m is no more than $O(log\ n)$ time the optimal solution. Their algorithm is based on partitioning the polygon into convex components. The convex components are views as sets and approximation set

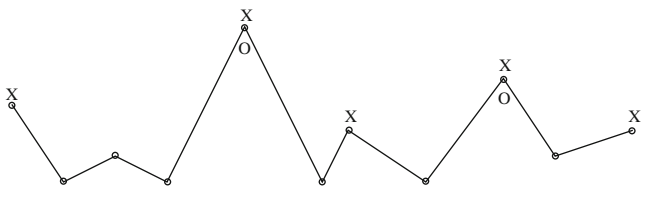


Fig. 75.2 Distinguishing one-sided and two-sided guarding

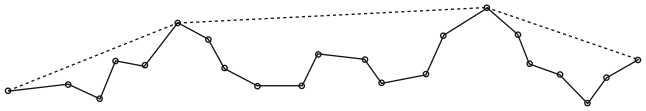


Fig. 75.3 Formation of pockets

covering algorithm is used to obtain approximation solution for art gallery problem. The time complexity of the algorithm is $O(n^4)$ for the simple polygon and $O(n^5)$ for the polygon with holes.

Ben-Moshe et al. [12] have reported an approximation algorithm for guarding 1.5D terrain by point guards. The idea is to process *pockets* of the terrain separately by evaluating the visibility of convex vertices in each packet. The *pockets* are formed when 1.5D terrain is enclosed by the convex hull boundary as shown in the Fig. 75.3.

In the figure, convex hull boundary (shown by dashed edges) and 1.5D terrain induce three pockets. The authors [12] made complicated case analysis to develop an approximation algorithm of constant factor for covering the terrain. The time complexity of the algorithm is $O(n^2)$ where n is the number of vertices in the terrain. In this algorithm it is not clear what is the exact value or bound of the constant factor.

75.3 Fast Heuristic for Covering 1.5D Terrain

75.3.1 Visibility Properties of 1.5D Terrain

We start with the description of the properties and characterization of 1.5D terrain needed for developing fast heuristics. Since the problem of placing a minimum number of point guards in 1.5D terrain is NP-Hard [7] it is motivating to come up with good heuristic methods that execute relatively fast in practical applications. Due to simpler structural properties of 1.5D terrain, guard placement is relatively easier compared to guard placement in a simple polygon.

Observation 75.1
For simple polygons, it is known that visibility of all vertices does not imply that all of the boundary is visible [1] This fact applies to 1.5D terrain as shown in Fig. 75.4.

In the figure, two guards are placed at v_1 and v_5 shown by the solid circles. All the remaining vertices v_2, v_3, v_4, v_6 and v_7 are visible from these two guards but the edge $<v_3, v_4>$ is not visible.

Observation 75.2
A guard placed at a non-convex vertex can be moved to the adjacent convex vertex without losing any coverage.

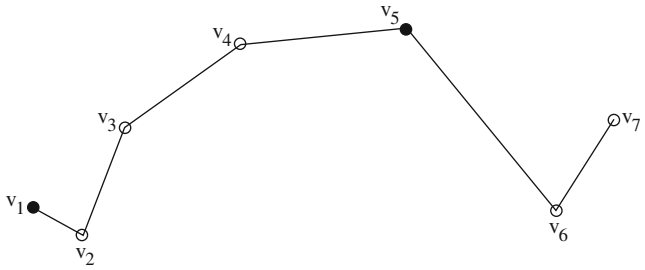


Fig. 75.4 Illustrating Observation 75.1

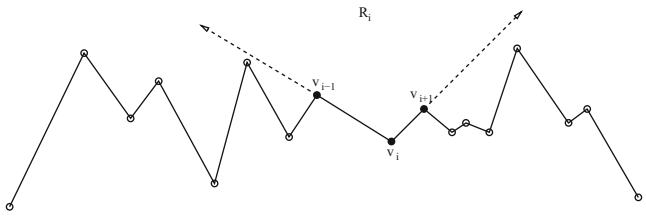


Fig. 75.5 Illustrating proof of Lemma 75.1

Definition 75.1 (Zig-Zag Terrain) A 1.5D terrain in which no two consecutive vertices are convex.

Lemma 75.1 *In a Zig-Zag 1.5D terrain, covering all vertices implies the covering of all boundary points.*

Proof Suppose there is an edge $e_i = (v_{i-1}, v_i)$ which is not completely visible. If a point guard is at v_{i-1}, v_i, or v_{i+1} then e_i is clearly visible. If v_i is visible from vertex v_j (other than v_{i-1} v_i, $v_{i+1)}$ then v_j must be in the sector R_j formed by v_{i-1}, v_i, v_{i+1} as shown by dashed rays in Fig. 75.5. Consequently, points of e_i are visible from that guard due to the convexity of the sector R_i.

75.3.2 Greedy Ranking Algorithm

This algorithm works in two stages: (i) greedy placement stage and (ii) redundancy removal stage. In the first stage, the next point guard is placed on the vertex that covers the maximum number of non-illuminated edges.

The location of vertex guards is determined based on their rank. The *rank* of a node v_i is the number of non-covered edges that can be covered by placing a point guard at v_i. Initially all edges are not covered, and the rank of each node is the number of edges visible from them. The initial rank of nodes is illustrated in Fig. 75.6a.

We describe the working of the algorithm with this running example. Since vertex v_{10}, v_{13}, v_{16}, v_{19} have the highest rank (5), we pick the vertex v_{16} arbitrarily and place the first guard at v_{16} (shown by a triangle symbol). Note that if more than one vertex has the same rank then we pick the vertex arbitrarily.

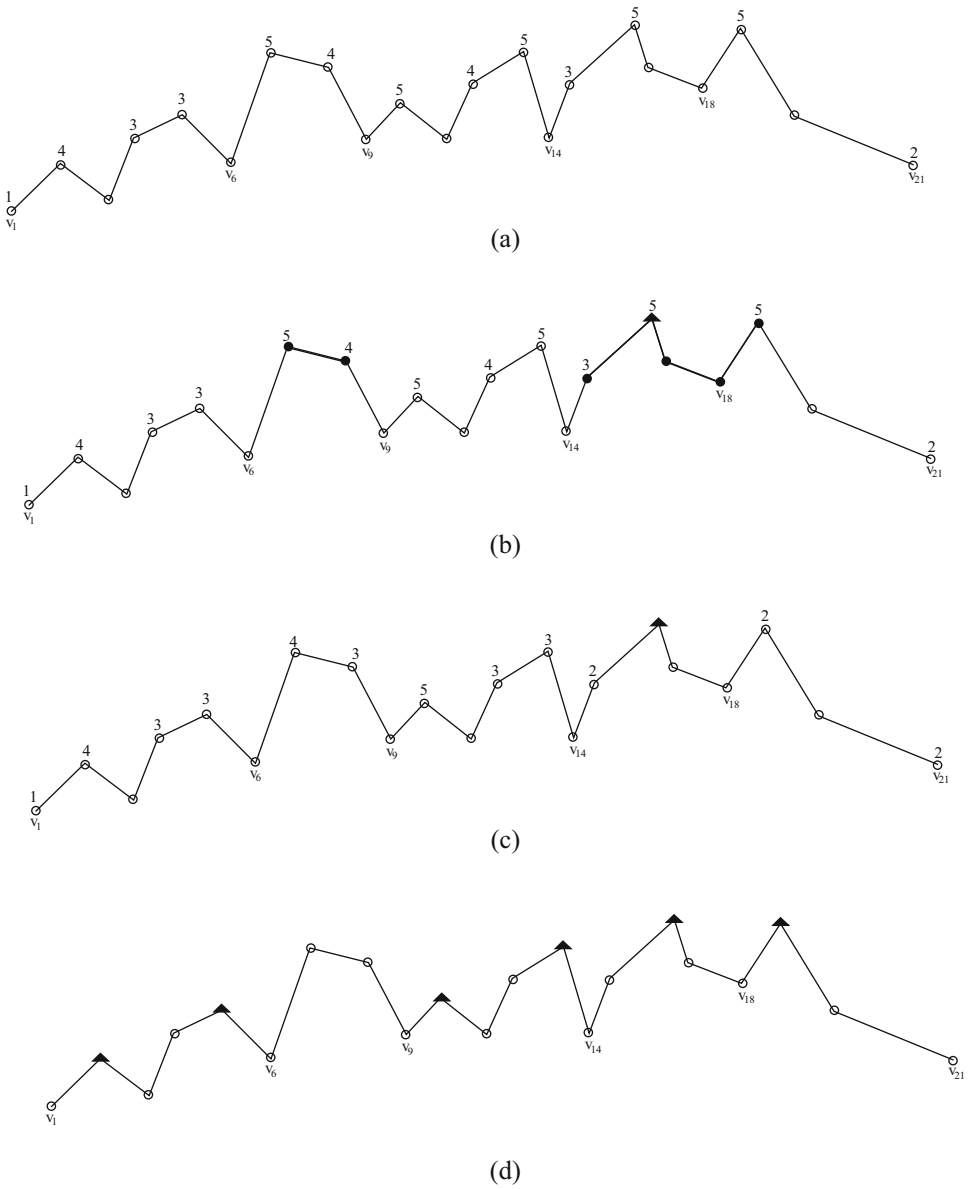


Fig. 75.6 Illustration of Greedy Ranking Algorithm. (**a**) Illustrating Initial Node Ranking. (**b**) Illumination state after placing the first guard. (**c**) Ranking of vertices after placing first guard. (**d**) Guard placement after completing first stage

After the placement of the guard at v_{16}, all the edges visible from v_{16} are determined (visible vertices are drawn with filled circle and visible edges are drawn with thick line in Fig. 75.6b). Based on this placement, the ranking of the nodes is recomputed. In recomputing the rank, only the uncovered edges are considered. The new ranks are shown in Fig. 75.6c.

It is noted that for the vertices where guards are already placed, the rank is not needed and not shown in Fig. 75.6c. In the second round of the ranking, vertex v_{10} has the highest rank and a guard is placed there. The process of re-ranking and guard placement is continued until all edges are covered. In our running example, the first stage is completed after 6 rounds of ranking. The placement of guards after completion of the first stage (greedy ranking) is shown in Fig. 75.6.

The greedy ranking algorithm places guards incrementally (one at a time) by identifying the vertex with highest rank. After each guard placement, the rank of the vertices are recomputed (updated) so that edges already covered are excluded in the ranking accumulation. The algorithm stops when all the edges are covered. A formal sketch of "Greedy Ranking Algorithm" is shown in Algorithm 1.

A straight forward implementation of the Algorithm 1 can takes $O(n^3)$ time in the worst case. Step 4 (ranking step) can be implemented by checking the intersection of possible edges with terrain edges which can take $O(n^2)$ time in the worst case. To implement Step 7 (marking covered edges), we can similarly check the intersection of candidate visibility edges with edges of terrain, which again takes $O(n^2)$ time. If the while loop repeats **n** times, then the total time complexity of the algorithm is $O(n^3)$. This time complexity is rather high. An improved implementation of Algorithm 1 based on the visibility graph is described next.

Algorithm 1 Greedy Ranking Algorithm

1. **Input** : Terrain chain Ch_1 $\{v_o, v_1, \ldots\ldots v_{n-1}\}$
2. **Output**: Subset S_g of Ch_1 where guards are placed
3. **while** all edges are not covered **do**
4. RankVertices (Ch_1 $\{v_o, v_1, \ldots\ldots v_{n-1}\}$)
5. u = getHighestRankedVertex (Ch_1)
6. u.guard = True
7. Mark edges covered by u
8. $S_g = \{v \mid v.guard = True\}$
9. Output S_g

function RankVertices (chain Ch_1 $\{v_o, v_1, \ldots\ldots v_{n-1}\}$)
for all v in Ch_1 **do**
if v is convex and no guard placed at v **then**
v.rank = Number of newly covered edges

The visibility graph in the presence of a polygonal object is defined by considering the polygonal boundary as obstacles.

The visibility graph in the presence of 1.5D terrain can be defined similarly and an example is shown in Fig. 75.7a. A visibility graph can be computed in time proportional to its size by using an algorithm given in [13] When the visibility graph is computed in [13] visibility edges emanating from a vertex are available in angularly sorted order around it. This structure of the output of the algorithm in [13] can be used to implement the Greedy Ranking Algorithm efficiently. Initial ranking of vertices can be obtained by simply reading off the number of visibility edges emanating from each vertex. When a guard is placed at a vertex, we can define the notion of Uncovered Visibility Graph (UVG) by considering only those visibility edges that are connected to the uncover edges of the terrain. When the first guard is placed (indicated by filled triangle sign above the terrain vertex), the resulting visibility graph is shown in Fig. 75.7b.

The efficient version of Algorithm 1 is based on updating UVG as guards are placed. Initially, UVG is given by the standard visibility graph. When a guard is placed at vertex v_i, visibility edges emanating from v_i and its incident vertices are deleted to update UVG. The updated rank of vertices are the counts of visibility edges corresponding to the updated UVG. In order to retrieve and update the ranks of nodes, the vertices of **UVG** are maintained in a priority queue Q, using the priority of the number of visibility edges emanating vertices. A formal sketch of the algorithm is listed as Algorithm 2.

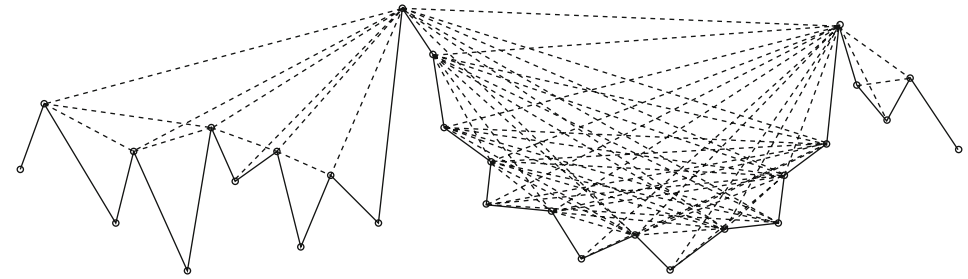
The time complexity of Algorithm 2 can be done as follows. Step 3 takes $O(|E|+|V|)$ [13]. One delete operation and decrease key operation can be done in $O(log|V|)$ time. Each execution of *while* loop removes at least one visibility edge to update UVG. Hence the whole loop executes at most $O|E|$ time. Thus, the total time complexity is $O(|E|log|V|)$

Algorithm 2 Improved Greedy Ranking Algorithm

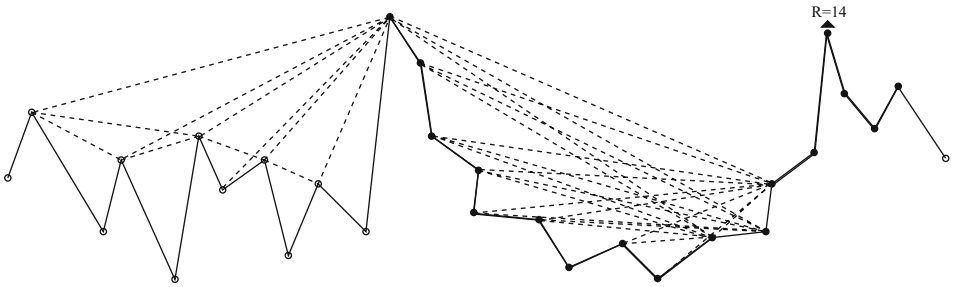
1. **Input**: Ordered list of vertices chain V $\{v_o, v_1, \ldots\ldots v_{n-1}\}$ of terrain **T**
2. **Output**: Subset S_g of Ch_1 where guards are placed
3. Compute Visibility Graph (VG) for T
4. Store Vertices of VG in max priority Queue Q in the priority of vertex degree
5. Set uncovered visibility graph UVG to VG
6. $S_g = \Phi$
7. **while** UVG contains edges **do**
8. v = Q.getMax()
9. $S_g = S_g\ U\ \{v\}$
10. Q.deleteMax()
11. **for** all vertices u adj to v
12. decrease key of u by 1
13. remove edge (u,v) from UVG
14. Output S_g

75.4 Conclusion

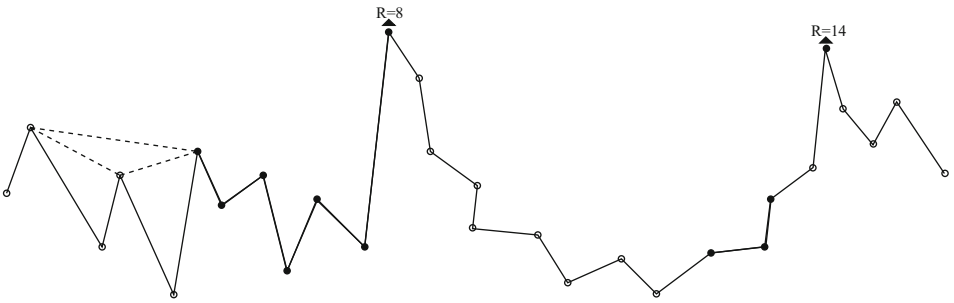
We presented a brief review of existing algorithms for placing point guards in 1.5D terrains and simple polygons. We presented a heuristic algorithm for covering 1.5D terrain by point guards. We have also developed another heuristic called "Greedy Forward Marching Algorithm". Experimental results of both heuristics are available in the full version of the paper. The second heuristic and the results of the experimental performance of both heuristics will be reported in the near future in appropriate outlets. Our experimental results show that the performance of the Greedy Ranking Algorithm is better than the performance of the Greedy Forward Marching Algorithm. We observed this result on several terrain input sizes 10, 25, 50, 75,...,7000. For all these input sizes, the data shows that the performance of Greedy Ranking is consistently better. One of the additional contributions of investigation (in full version of the paper) is the generation of 1.5D terrain data of various sizes. The generation is done randomly by using a *guiding strip*. At present, an implementation by the guiding strip is taken as a shape with zig-zag structure. To make it more realistic it would be interesting to have strips of other structures. This can be an interesting future work. The performance of both heuristics can be improved. One approach for improvement would be to look forward beyond the Next Candidate Node while placing the next guard. This is expected to improve the performance of the algorithm at the expense of time complexity. Recently, some authors have proposed the notion of two-sided guard placement [10]. It would be interesting to convert our proposed heuristics to a two-sided version of visibility.



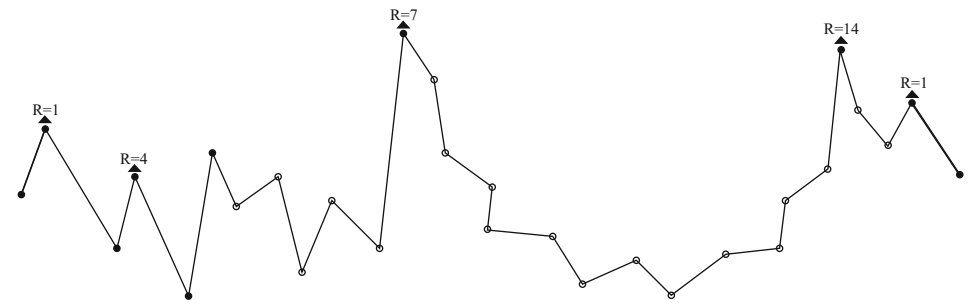
(a)



(b)



(c)



(d)

Fig. 75.7 Illustration of visibility graph. (**a**) Illustration of visibility graph. (**b**) Illustration of visibility graph for uncovered edge after placing first guard. (**c**) Illustration of visibility graph for uncovered edge after placing second guard. (**d**) Illustration of visibility graph for uncovered edge after placing all guards

There is ample scope to developing better algorithms for identifying redundant guards. In the method proposed to identify redundant guards (described in the full version of the paper) at v_i we only look for the pair of guards (one to the left and one to the right of v_i). A generalization of this technique is to look for coverage by more than two guards (say three). This should improve the spotting of redundant guards at the expense of time complexity.

A better approach for generating realistic 1.5D terrain would be to sample points on the horizon of a real terrain and connect them. This approach is certainly feasible and would be a good avenue for further research. We have taken an unlimited visibility model for defining visible vertices: two vertices are visible as long as the line segment connecting them does not intersect with the terrain, no matter how far apart they are located. A more realistic model is to incorporate the notion of *limited visibilit*y. Under this model, two vertices v_i and v_j are visible if (i) the line segment connecting them does not intersect with the terrain, and (ii) they are not farther apart than a certain distance d. It would be an interesting research exercise to develop guard placement algorithms under limited visibility.

References

1. O'Rourke, J.: *Computational geometry in C*. Cambridge University Press, Cambridge (1998)
2. Gewali, L.P.: Recognizing s-star polygons. Pattern Recogn. **28**(7), 1019–1032 (1995)
3. Vassilev, T., Pape, S.: Visibility: finding the staircase kernel in orthogonal polygons. Essay Math Stat. **2**(05), 79–89 (2012)
4. O'Rourke, J.: *Art gallery theorems and algorithms*, vol. 57. Oxford University Press, Oxford (1987)
5. De Berg, M., Van Kreveld, M., Overmars, M., Schwarzkopf, O.C.: Computational geometry. In: *Computational geometry*, pp. 1–17. Springer, Berlin (2000)
6. Lee, D., Lin, A.: Computational complexity of art gallery problems. IEEE Trans Inform Theory. **32**(2), 276–282 (1986)
7. King, J., Krohn, E.: Terrain guarding is np-hard. SIAM J Comput. **40**(5), 1316–1339 (2011)
8. Garey, M.R., Johnson, D.S.: *Computers and intractability*, vol. 29. Freeman, New York (2002)
9. Gewali, L., Dahal, B.: Algorithms for tower placement on terrain. Adv Intell Syst Comput. **36**(6), 551–556 (2019)
10. Lai, W.-Y., Hsiang, T.-R.: Continuous terrain guarding with two-sided guards. CCCG. **2018**, 247–252 (2018)
11. Ghosh, S.K.: Approximation algorithms for art gallery problems in polygons(report). Discr Appl Math. **158**, 6 (2010)
12. Ben-Moshe, B., Katz, M., Mitchell, J.: A constant-factor approximation algorithm for optimal terrain guarding. In: Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, ser. SODA '05. Society for Industrial and Applied Mathematics, pp. 515–524 (2005)
13. Ghosh, S.K., Mount, D.M.: An output sensitive algorithm for computing visibility graphs. In: 28th Annual Symposium on Foundations of Computer Science (SFCS 1987), pp. 11–19 (1987)