Maximizing the Guarded Boundary of a Dynamic Art Gallery

Mihir Patel

May 9, 2025

Abstract

An abstract.

1 Introduction

We consider a variant of the classic Art Gallery problem, where we instead seek to optimize the length of the boundary seen by the guards, not the number of guards themselves. Specifically, given a simple polygon P and $k \in \mathbb{N}$, we want to find the k vertex guards which maximize the length of the boundary of P that is watched. We define the set of vertices of P as V_P, and $L(S)$ as the length of the boundary seen by the set of vertex guards/vertices S. Note that $L(S)$ is necessarily at most the perimeter of P.

Maximum Length Vertex Guard

Input: A simple polygon P and a positive integer $k \in \mathbb{N}$.
Task: Find a set of vertices $S \subseteq V_P$ of size at most k such that $L(S)$ is maximized.

We also consider a weighted variant of this problem, where P is a simple polygon composed of (possibly collinear) weighted line segments. Art galleries contain paintings, and some have more value than others. With limited guards, a realistic task would be to maximize not the total boundary watched, but the total value of paintings watched. In our definition, a weighted segment must be completely seen by our k guards to be considered "watched". We define $W(S)$ as the weighted analog of $L(S)$, the sum of weights of all segments on the boundary of P that are completely watched by guards in S.

Maximum Value Vertex Guard

Input: A weighted polygon P and a positive integer $k \in \mathbb{N}$.
Task: Find a set of vertices $S \subseteq V_P$ of size at most k such that $W(S)$ is maximized.

Finally, we consider a dynamic version of Maximum Value Vertex Guard, which we'll call Dynamic Maximum Value Vertex Guard. New paintings may arrive in an art gallery, and their arrangement around the gallery may change. Thinking of vertex guards as cameras, it can be costly and time-consuming rearrange and reinstall cameras across the room (at least compared to moving the paintings around). If the segment weights on the weighted polygon change, how can we modify our solution (with minimal changes) to maintain an approximate solution for Maximum Value Vertex Guard at each timestep?

Obviously this dynamic problem needs to be specified a lot more, but hopefully the general idea of/motivation for the problem is showing through. I am trying to create a dynamic set cover analog for art gallery, where elements are inserted/removed one at a time, and the goal is to maintain a set cover that is still an approximate solution (without simply recalculating a set cover at each change).

1.1 Related Works

Need a paragraph discussing more conventional Art gallery problem literature (authors+results). Klee, O'Rourke, Das

In [4, 7, 6], Fragoudakis et al. pose both Maximum Length Vertex Guard and Maximum Value Vertex Guard. They prove that both problems are APX-complete in [7], meaning these problems are NP-hard and also permit no PTAS unless $P = NP$. In [4], they present a $(1 - 1/e)$-approximation for maximizing the vertex-guarded *interior* of a polygon which runs in $O(k^2n^2)$ and depends on segmenting the polygon into visibility regions.

[1] presents several inapproximability results for art gallery problems, with α-Floodlights.

Need to discuss some more dynamic set cover. Here is the most recent paper I've been able to find.

[2]

1.2 Our Contributions

For Maximum Length Vertex Guard and Maximum Value Vertex Guard, we extend the results from [4] to get a $(1 - 1/e)$-approximation, that runs in $O(kn^2)$. We use the monotonicity and submodularity of our objective functions L and W to get this bound, presenting a simpler (and slightly more efficient) approach than the Finest Visbility Segmentation approach from [4, 6]. We also prove several inapproximability results for these problems.

For Dynamic Maximum Value Vertex Guard, we analyze the performance of several strategies, showing some of them may perform arbitrarily badly compared to an optimal solution, while some achieve a bound.

We consider two problems proposed, and introudce a new problem. We focus on the weighted version.

2 Submodularity of Maximum Value Vertex Guard

In this section, we prove that the objective functions of both Maximum Length Vertex Guard and Maximum Value Vertex Guard are monotone and submodular. This then implies that greedily maximizing the objective will yield a $(1 - \frac{1}{e})$ approximation for both problems. We also provide a brief analysis of the running time of these greedy algorithms, which is an improvement on the approximation algorithms proposed for these problems in [4, 7, 6].

We start with the objective function of Maximum Length Vertex Guard, L. Recall that given a simple polygon P and a set of vertex guards $S \subseteq V_P$, $L(S)$ denotes the length of the guarded boundary of P, and is necessarily less than or equal to the permiter of P. It is fairly immediate to see that L is monotone.

Observation 1. *For any $S \subseteq T \subseteq V$, $L(S) \leq L(T)$.*

Proof. Consider a point p on the portion of the boundary of the polygon counted by $L(S)$, then p must be seen by some $v \in S$. As $S \subseteq T$, $v \in T$ as well, implying p is counted by $L(T)$. Every point on the boundary counted by $L(S)$ is also counted by $L(T)$, and thus $L(S) \leq L(T)$. □

We can also show that L is submodular with a little bit more work.

Claim 1. *For any $S \subseteq T \subseteq V$ and $v \notin T$, $L(S \cup \{v\}) - L(S) \geq L(T \cup \{v\}) - L(T)$.*

Proof. First observe that, by Observation 1, $L(S \cup \{v\}) - L(S) \geq 0$, meaning this claim is trivial if $L(T \cup \{v\}) - L(T) \leq 0$. So we only consider the case where $L(T \cup \{v\}) - L(T) > 0$, or in other words, there is some portion of the boundary of the polygon guarded by $T \cup \{v\}$, but not T. Then for every point p in this portion, p is visible to v, but not T. As $S \subseteq T$, by Observation 1, p is visible to S either.

In short, we know that p is visible to $S \cup \{v\}$ but not S, and as this is true for every point on the boundary seen by the set of guards $T \cup \{v\}$, but not T, $L(S \cup \{v\}) - L(S) \geq L(T \cup \{v\}) - L(T)$, as desired. □

These two facts also lift quickly to the objective function of Maximum Value Vertex Guard, W. Given a simple polygon P made up of weighted segments and a set of vertex guards $S \subseteq V_P$, $W(S)$ denotes the total weight of the guarded boundary of P. Note that a weighted segment must be *completely* visible to a set of guards to be considered "guarded" and contribute to the total weight. The proofs of monotonicity and submodularity for W proceed exactly the same as those of L (we only need to swap L and W).

Observation 2. *For any $S \subseteq T \subseteq V$, $W(S) \leq W(T)$.*

Claim 2. *For any $S \subseteq T \subseteq V$ and $v \notin T$, $W(S \cup \{v\}) - W(S) \geq W(T \cup \{v\}) - W(T)$.*

Equipped with these properties of L and W, we can quickly derive greedy algorithms for Maximum Length Vertex Guard and Maximum Value Vertex Guard. Start with an empty solution set S. While the size of S is less than k, find the vertex v which provides L with the most marginal gain. That is, the vertex v which maximizes $L(S \cup \{v\}) - L(S)$. Then add v to S and repeat, until $|S| = k$. Then, because we L and W are both monotone submodular, we get the following facts about maximizing monotone submodular functions under cardinality constraints (i.e. the type of problem Maximum Length Vertex Guard and Maximum Value Vertex Guard both are).

Theorem 1 ([3])**.** *The greedy algorithm is a $(1-1/e)$-approximation algorithm for Maximum Length Vertex Guard.*

Theorem 2 ([5])**.** *For any $\epsilon > 0$, there is no $(1 - 1/e + \epsilon)$-approximation algorithm for Maximum Length Vertex Guard, unless* P=NP*.*

And the same bounds are true for the greedy algorithm which maximizes W for Maximum Value Vertex Guard. These algorithms require k iterations, and in a given iteration must check the marginal gain provided by at most n vertices. To check the marginal gain of a given vertex v, we can maintain the boundary guarded by S (and its length), calculate the boundary guarded by v, and then take the union of both boundaries. Finding the boundary guarded by a vertex can be done in $O(n)$ [8], so with proper data structure management, one "marginal gain check" can be done in $O(n)$. Thus the greedy algorithm has worst case running time of $O(kn^2)$.

We've shown that using the monotonicity and submoduarity of our objective functions, we can achieve simple greedy algorithms for Maximum Length Vertex Guard and Maximum Value Vertex Guard. While [7, 4, 6] propose algorithms that also achieve $(1 - 1/e)$ approximations for these problems, we also show that these are the *best possible* approximations for these problems, unless P=NP. Our algorithms also run in $O(kn^2)$ time, while their algorithms run in $O(n^4)$ and $O(k^2n^2)$, both less efficient than ours.

3 Intuition-Building Examples for Budgeted Maximum Value Vertex Guard

In this section, we show that three natural greedy strategies for Budgeted Maximum Value Vertex Guard may all perform arbitrarily badly compared to an optimal solution.

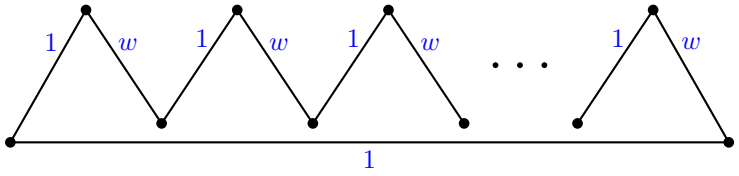


Figure 1: apex vertices have cost w, everything else has w^2, budget is w^2.

In other words, these strategies will not automatically form approximations for Budgeted Maximum Value Vertex Guard, as there exists a problem instance where the total weight they achieve is *not* boundedly close to the optimal. Note that this is in contrast to Maximum Length Vertex Guard and Maximum Value Vertex Guard, where the natural greedy strategy formed a constant-factor approximation, validating the added difficulty of Budgeted Maximum Value Vertex Guard.

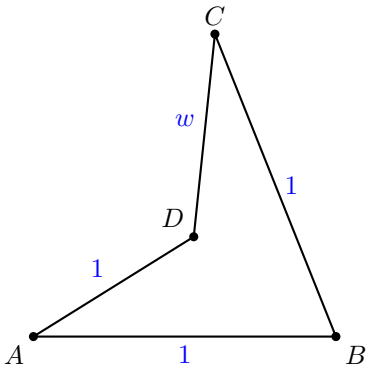
A first attempt could be to greedily add the vertex which maximizes the total weight guarded. To see why this fails, fix a $w \in \mathbb{N}$ such that $w \geq 3$. Then construct the polygon depicted in Figure 1, with w many "cones". The right edge of each cone has weight w, while all other edges have weight 1. The apex of each cone has cost w, while all other vertices have cost w^2. The budget B is equal to w^2, and now we have an instance of Budgeted Maximum Value Vertex Guard, with Figure 1 and $B = w^2$.

Now the greedy strategy would start by choosing the vertex that maximizes the total weight guarded. Initially, this is one of the "valleys" in polygon, as these vertices guard $2w+2$ total weight, whereas all other vertices guard at most $w+1$. Any one of these vertices has cost w^2, so adding it to our solution uses our entire budget. Thus the algorithm which greedily maximizes total weight guarded ends up guarding $2w+2$ total weight on Figure 1.

However, an optimal strategy would be to place a guard at each apex. There are w many apices and each one has cost w, so to use all of them would cost w^2 (which is exactly our budget). This guard placement also guards our entire polygon boundary, meaning the total weight guarded by the optimal set of guards is $w(w) + w(1) + 1 = w^2 + w + 1$. To compare total weight achieved between these two algorithms, the weight achieved by the greedy divided by the weight achieved by the optimal is equal to $\frac{2w+2}{w^2+w+1}$. We can make this value arbitrarily small as we increase w, meaning the greedy algorithm which maximizes total weight guarded performs arbitrarily badly compared to an optimal solution.

Now we consider two alternative greedy strategies and show that they also fail, using Figure 2. The first would be to greedily add the vertex with cheapest cost. The second would be to greedily add the vertex which maximizes the ratio of marginal gain in total weight to vertex cost. Starting with the first strategy, consider Figure 2 where weights and costs are displayed, and the budget is $w+4$. The cheapest vertex is A, which has cost of 2. The vertex A guards total weight of 2, and once we have placed a guard there we cannot afford to place guards at any of the remaining vertices. Thus this greedy choosing the cheapest vertex achieves total weight of 2 on this example. An optimal strategy would be to place a guard on either D or B, this stays within budget and guards the entire polygon, achieving total weight of $w+3$. Comparing the weight achieved by each strategy, we have $\frac{2}{w+3}$, which we can make arbitrarily small as we increase w.

Similarly, if we wanted to greedily choose the vertex which maximizes the ratio of marginal gain in total weight to vertex cost (i.e., provides the best "bang-for-your-buck"), we would choose vertex A. It guards weight of 2 and costs 2, giving a ratio of 1. Any other vertex



Vertex	Cost
A	2
B	$w+4$
C	$w+4$
D	$w+4$

Figure 2: caption

can guard at most $w+3$, while costing $w+4$, and $\frac{w+3}{w+4} < 1$ for all w. Again, choosing A prevents adding any more guards, achieving a total guarded weight of 2, while an optimal strategy could guard $w+3$, and $\frac{2}{w+3}$ can be arbitrarily small for large w.

4 Conclusion

References

[1] A. Abdelkader, A. Saeed, K. Harras, and A. Mohamed. The inapproximabiliy of illuminating polygons by α-floodlights. *Proceedings of the Canadian Conference on Computational Geometry 2015*, August 2015.

[2] A. Bukov, S. Solomon, and T. Zhang. Nearly optimal dynamic set cover: Breaking the quadratic-in-f time barrier. pages 824–863, 2025. doi: 10.1137/1.9781611978322.24.

[3] cornuejols.

[4] I. Emiris, C. Fragoudakis, and E. Markou. Maximizing the guarded interior of an art gallery. *European Workshop on Computational Geometry*, pages 165–169, March 2006.

[5] feige.

[6] C. Fragoudakis, E. Markou, and S. Zachos. How to place efficiently guards and paintings in an art gallery. *Lecture Notes in Computer Science*, 3746:145–154, November 2005. doi: 10.1007/11573036_14.

[7] C. Fragoudakis, E. Markou, and S. Zachos. Maximizing the guarded boundary of an art gallery is apx-complete. *Computational Geometry*, 38:170–180, October 2007.

[8] visibility.