

Computing Multiple Watchman Routes ^{*}

Eli Packer[†]

Abstract

We present heuristics to compute multiple watchmen routes. Given a polygon (with or without holes) and a parameter k , we compute a set of k routes inside the polygon such that any point inside the polygon is visible from at least one point along one route. We measure the quality of our solutions by either the length of the longest route and the sum of route lengths, where the goal is to minimize each. We start by computing a set of static guards [2], construct k routes that visit all the static guards and further shorten the routes while maintaining full coverage of the polygon. We implemented the algorithm and present extensive results to evaluate our methods, including a comparison with lower bound routes based on the idea of visiting large number of visibility-independent “witness points”. Our experiments showed that for a large suite of input data our heuristics give efficient routes that are comparable with the optimal solutions.

1 Introduction

The *Art Gallery* problem is a famous computational geometry problem that has been extensively studied in the previous three decades. Presented in 1973 by Klee, the idea is to place a minimum number of point guards that collectively cover the interior of a given simple polygon¹. Since then, numerous variants have been proposed and studied. Some of the most important results were the hardness proofs of the basic variant (point guards inside a simple polygon) [14] and others. These motivated the studying of approximations and heuristics.

As opposed to the basic variant which deals with static guards, the *watchman route* variant is concerned with guards that can translate along routes. The goal is similar: make any point inside the polygon (or any other domain) visible by at least one point along one of the routes. This problem is motivated by many applications that are involved in elements such as security and surveillance (e.g., guarding, exploring and analyzing buildings and areas), saving energy and time (e.g., it takes less frames to photograph on area), efficient simulations and more. In this problem, as the number of guards is usually predetermined, the measure of the result is often the route lengths (the Euclidian metric is usually used here, but other metrics, such as the number of links in the routes, have been used too). Interestingly, the minimum watchman route (one guard) inside a simple polygon has an exact polynomial time solution [20]. Unfortunately, extending the problem to support holes inside the polygon or allowing more than one guard (when minimizing the longest route) make the corresponding decision problems hard (the hardness proofs use simple reductions from the TSP [8] and partition [15] problems respectively).

^{*}Work on this paper has been partially supported by the National Science Foundation (CCR-0098172, CCF-0431030). The author thanks Esther Arkin, Alon Efrat, Matthew Katz, Joseph Mitchell, Girishkumar Sabhani and Valentin Polishchuk for interesting and helpful discussions on related problems.

[†]Department of Computer Science, Stony Brook University, Stony Brook, NY 11794-4400. epacker@cs.sunysb.edu.

¹In the terminology of the art gallery study, point guards can see in any direction with no distance limit. A guard is said to cover a point inside the polygon if the line segment connecting them does not intersect any edge of the polygon.

We study the *k-watchman routes* in which $k \geq 1$, where multiple watchmen are allowed to translate inside a polygon (possibly with holes). Two natural goals in this case are to minimize the maximum length of any route and to minimize the sum of the route lengths. We denote the corresponding problems by $KWRP_m$ and $KWRP_s$ respectively ($KWRP$ stands for *k-watchman routes* inside a polygon). We note that these two problems are well motivated: one motivation behind $KWRP_m$ is to minimize the time it takes to cover the polygon while the motivation of $KWRP_s$ could be to save the total energy or frames taken by the complete system. We show later that it is even hard to give any meaningful proven approximation to both measures.

Two points in a polygon P are termed *independent* if there is no point $p \in P$ which covers both. Thus, a set of i independent points form a lower bound i on the polygon static guarding number. It follows that a set of k (for either $KWRP_m$ and $KWRP_s$) that optimally cover an independent set forms a lower bound on the optimum solution. We use this idea to evaluate the efficiency of some of our experiments (see Section 5).

Our Contribution. We propose heuristics for computing watchman routes that cover polygons (with or without holes), for both $KWRP_m$ and $KWRP_s$. While it is impossible to develop exact or even approximate polynomial time algorithms for both problems (unless $P = NP$), we conduct an extensive experimental analysis of their performance. We show that our heuristics work well in practice for many kinds of polygons, and in some cases compare to lower bounds obtained by the idea of independent witness point set. As far as we know, this work is the first attempt to conduct a systematic experimentation with watchman route heuristics.

As we mentioned above, the 1-watchman route problem has been optimally solved with a polynomial time algorithm. However, to the best of our knowledge, neither implementation nor experiments have ever been reported. Perhaps the reason is that the algorithm is not easy to implement. Further, it is not clear if this algorithm will suffer from robustness problems. Hence, this work is also targeted for this specific and important variant, as when we set $k = 1$ we compute the 1-watchman route.

Related Work Good references for the art gallery various problems are [17, 19, 22]. Two detailed reports that provide valuable information about the watchman route problem and present algorithms for restricted versions are [16, 17]. The 1-watchman route problem has been extensively studied. After a few publications that were found to have flaws, Tan et al. [21] gave an $O(n^4)$ time algorithm where the starting point is given and finally Tan [20] presented an $O(n^5)$ algorithm for the general case which followed [5]. Other interesting variants that have been studied are the minimum-link watchman route [1, 3], Pursuit-Evasion [18] and others [6, 11, 13, 23].

The rest of this paper is organized as follows. In the next section we provide background information. In Section 3, we present our algorithm. Details about our implementation are given in Section 4. In Section 5 we present our experiments with the software we have implemented. We devote Section 6 to describe a few ways for future improvements and conclude in Section 7.

2 Preliminaries

Given a polygon P (possibly with holes) and a point $p \in P$, we denote by $\mathcal{V}(p)$ (P is omitted for simplicity) the set of points inside P that are visible from p .² It is easy to observe that $\mathcal{V}(p)$ is a star-shaped polygon

²Two points are visible to each other if the line segment that connects them does not intersect any edge of the polygon.

and it is termed the *visibility polygon* of p . A set of points $S \subset P$ is said to cover P if $\bigcup_{p \in S} \mathcal{V}(p) = P$. The classic art gallery problem is to find a smallest such set.

Over the years numerous variations of this problem have been proposed and studied. In one interesting family of the variations, guards are allowed to translate inside the polygon along predefined routes. In this case, the guards are often termed *watchmen* or *mobile guards* and their routes are termed *watchman routes*. Let w be a watchman with route R_w inside a polygon P (P is omitted for simplicity). Let $\mathcal{V}(w) = \bigcup_{p \in R_w} \mathcal{V}(p)$ be defined similarly to the visibility polygons of the static guards. The goal here is to cover P as well, namely to find a set of watchmen S such that $\bigcup_{w \in S} \mathcal{V}(w) = P$. In this context, the size of S is usually given (we denote it by k) and the measure (or quality) of the solution involves the length of the routes. Two popular measures are the length of the longest route (we denote the corresponding problem by $KWRP_m$) and the sum of route length (denoted by $KWRP_s$). More formally, the measure of $KWRP_m$ and $KWRP_s$ are $\max_{w \in S} L(w)$ and $\sum_{w \in S} L(w)$ where $L(w)$ is the length of route R_w . Table 3 summarizes the complexity of computing watchman routes for simple polygons. Table 2 summarizes results for constrained polygons that were established in [16].³ Both tables are borrowed from [16], while we updated the the complexity of the MinSum variant in Table 3 for two or more guards. We note that when the polygon may have holes, both MinSum and MinMax become NP-hard for any number of watchmen.

Optimization criterion	Number of watchman routes			
	1	2	...	arbitrary
MinSum	P	NP-hard	NP-hard	NP-hard
MinSum	P	unknown	unknown	NP-hard

Table 1: Complexity of computing sets of watchman routes of various sizes inside a simple polygon.

Optimization criterion	Polygon classes				
	Spiral	Histogram	Alp	Street	Simple
MinSum	P	P	P	NP-hard	NP-hard
MinSum	P	P	unknown	NP-hard	NP-hard

Table 2: Complexity of computing sets of watchman routes of any size, for some classes of polygons.

Next we show that the related decision problems of $KWRP_m$ and $KWRP_s$ cannot have any reasonable approximation (unless $P = NP$).

Theorem 2.1 *$KWRP_m$ and $KWRP_s$ can have no polynomial approximation algorithms unless $P = NP$.*

Proof: Suppose there is such a polynomial algorithm \mathcal{A} (for either $KWRP_m$ or $KWRP_s$) with running time $O(\Gamma)$. We show how to solve the basic art gallery problem (denoted by \mathcal{AG}) optimally in polynomial time using \mathcal{A} , thus contradicting the above assumption, unless $P = NP$. It can be easily verified that P can be guarded by k static guards if and only if \mathcal{A} returns routes of zero length, given k as a parameter. Given a polygon P with n vertices, the art gallery theorem states that $\lfloor \frac{n}{3} \rfloor$ guards are always sufficient to guard P . It follows that the solution to \mathcal{AG} can be found by searching the minimum k for which the solution of \mathcal{A} contains only routes of zero length. It would require $O(\log(n)\Gamma)$ time. It follows that unless

³In this table two kinds of polygons are defined as follows: (1) A polygon is an *alp* if it is monotone and one of the chains in the partition is a line segment parallel to the x -axis. (2) A polygon is a *street* if its boundary can be partitioned into two chains, each of which are guard sets for the polygon.

$P = NP$, such an approximation algorithm cannot exist. □

Given a polygon P and two points $p_1, p_2 \in P$, we say that p_1 and p_2 are *independent* if there is no point $g \in P$ such that both $p_1 \in \mathcal{V}(g)$ and $p_2 \in \mathcal{V}(g)$. Let S be a set of pairwise independent points in P of size m . It follows that m static guards are necessary (but may not be sufficient) to guard the points of S . Hence, computing independent sets is a convenient tool to find lower bounds for the art gallery problem and used in [2] for that purpose. We use an analogous idea in our work to compute lower bounds. Recall that k is the number of watchmen. By considering all partitions of S into k groups, and then computing the routes that cover all of the points in each group while optimizing the given problem (either $KWRP_m$ or $KWRP_s$), we can find lower bounds. Although we did not design or implement any heuristic to carry out this task in this work, we use this idea to find lower bounds, and use them in our experimental evaluation.

3 Algorithm

We next present a high-level description of our heuristics, followed by a more detailed description.

COMPUTE WATCHMAN ROUTES

Input: A polygon P (possibly with holes), k (number of watchmen) and an indication whether to perform $KWRP_m$ or $KWRP_s$

Output: A set of k watchman routes inside P that cover its interior

Measure: The length of the longest route (for $KWRP_m$) or the sum of length of the routes (for $KWRP_s$)

- (a) Compute a static guard set S with one of the proposed heuristics (A_1) of [2]. (A_1 is one of the three proposed heuristics, that was found efficient in time and produced good results.)
- (b) Build the visibility graph \mathcal{U} of $S \cup V$, where V is the set of vertices of P .
- (c) Using \mathcal{U} , compute the pairwise shortest paths between any $s_1, s_2 \in S$ inside P (denoted by \mathcal{Z}).
- (d) Build the minimum spanning tree of S inside P (denoted by \mathcal{T}) where the distance between any pair of points is computed from \mathcal{Z} .
- (e) Split \mathcal{T} into k subtrees, $\mathcal{T}_1 - \mathcal{T}_k$.
- (f) From this step we work on each subtree independently. We build a Hamiltonian route (let \mathcal{R} denote an arbitrary one) from each subtree $\mathcal{T}_1 - \mathcal{T}_k$, following the method by Christofides [9].
- (g) Substitute vertices along \mathcal{R} with others that shorten its length.
- (h) Remove redundant vertices of \mathcal{R} by connecting adjacent vertices, to shorten its length (we say that a vertex is redundant if when we connect its two adjacent vertices with the shortest path that connects them, the polygon remains fully covered).

We continue with a detailed description of the steps that require further explanation.

(a) We compute a static guard set S [2]. The idea is that routes cover the polygon if they visit all of the loci of the static guards. As we show in steps **g** and **h**, we may visit other vertices or skip them, but still guarantee that the coverage is fully maintained.

(e) We split \mathcal{T} into k subtrees, $\mathcal{T}_1\text{-}\mathcal{T}_k$. If the problem is $KWRP_s$ we do it by simply removing the longest $k - 1$ edges of \mathcal{T} . The heuristic is much more involved if the problem is $KWRP_m$. By a reduction from the partition problem [15], the corresponding optimization problem is hard. We partition \mathcal{T} by removing edges. The edges we choose split \mathcal{T} in order to minimize the weight of the heaviest subtree. We remove edges by using ideas from parametric search (see Section 4 for details).

However, in some cases we get better results if we make a different kind of split which we call *split around a vertex*. Consider the polygon in Figure 1.⁴ The results of our heuristic (without the optimization we describe here) for $KWRP_m$ are shown in Figure 1(a) and (b) for the minimum spanning tree and the output respectively. Figure 1(c) shows a different way to split \mathcal{T} . In this case it is split around the top vertex to induce the two subtrees in green and magenta. Working on these trees separately will produce the two routes depicted in Figure 1(d). Consider the generalization of this example to any number of very long ‘legs’ of the same length (denoted by x) and k guards. $KWRP_m$ produces the following results. For any $k > 1$, it removes $k - 1$ legs and remains with a tree that results in a route of length roughly $(m - k + 1)x$ where m is the number of legs. On the other hand, an optimal solution is obtained by splitting around v to subtrees each of which with $O(\lceil m/k \rceil)$ legs, resulting in a maximum route of length $\lceil m/k \rceil x$. It follows that our heuristic converges to an approximation of factor $\Theta(k)$ to the optimum.

In order to decrease this ratio, and remain simple at the same time, we use the following method which modifies \mathcal{T} . In case there is a vertex $v \in \mathcal{T}$ of high degree whose neighbors have low degrees, we remove most of the adjacent arcs of v and rebuild \mathcal{T} while not choosing any edge that connects v again. Our rule of thumb is to do so on a vertex if its degree is at least $\max(k/2, 5)$ larger than at least $d/2$ of its adjacent vertices where d is the degree of v . In this case we remove all adjacent arcs but $\lfloor \frac{2d}{\max(k/2, 5)} \rfloor$. This method improves the results in the above example significantly, giving the tree and routes depicted in Figure 1(e) and (f) respectively. Of course, this method can usually be improved by tuning (automatically or manually) our rule of thumb.

Figure 1(g) shows another interesting example where an optimum is achieved by adding a Steiner vertex (the black disc in the figure) and splitting around it.

We note that generalizing our heuristics by adding Steiner vertices and splitting around vertices in more complicated cases (while continuing to support the ideas that we use, namely removing edges), seems like a very challenging task which can be simply observed as a hard problem. It is still not clear to us how to develop efficient heuristics that will work together with our other ideas, and it is not clear how useful and time costly they will be. We hope to investigate this idea further in the future.

(g) We substitute some of the vertices along \mathcal{R} with others that shorten its length.

The vertices of \mathcal{R} can be partitioned into two groups: the first (denoted by \mathcal{R}_g) belong to the static guard set and the second (denoted \mathcal{R}_a) are the rest, namely the vertices along the paths that connect two vertices of \mathcal{R}_g . The idea here is to replace vertices of \mathcal{R}_g by others in order to shorten the length

⁴Note that some of the routes in this figure and others are not simple close chains. Some routes or parts of them sometime degenerate to polygonal chain or contain chains that connect two parts of the routes. The idea is that the watchman walks on them in both directions. It follows that some of the vertices along the routes may be of degree $d > 2$. In order to remove any possible ambiguity, we use arrows (here in Figure 1(f)) to clarify. In Figure 1(b) one of the watchman route degenerates to a single point (the watchman essentially becomes static). We represent it with a disc around the guard and do the same in other figures as well.

of \mathcal{R} . For any $p_1, p_2 \in P$, let $\alpha(p_1, p_2)$ be the shortest path from p_1 to p_2 inside P . Let $v \in \mathcal{R}_g$ and let $u, w \in \mathcal{R}_g$ be the two vertices before and after v along \mathcal{R} (excluding the vertices of R_a ; Note that if $|R_g| = 2$ then $u = w$). Let $z \in P$ be some point. The *replacement* of v by z (denoted by $R(v, z)$) is defined as modifying \mathcal{R} by removing $\alpha(u, v)$ and $\alpha(v, w)$ from \mathcal{R} and inserting $\alpha(u, z)$ and $\alpha(z, w)$ instead. Note that \mathcal{R} remains a closed curve after performing this operation.

Given a polygon P , let H be the set of reflex angles of P . We extend the edges that are adjacent to the angles of H into the interior of P , until they hit the boundary of P (denoted by $@P$; See an example in Figure 2(g)). We denote these extensions by Q . Let \mathcal{G} be the arrangement of $(@P) \cap Q$. We call \mathcal{G} the *extension arrangement* of P . For each vertex $v \in \mathcal{R}$, let $f(v)$ be the face of \mathcal{G} that contains it (if v is a vertex of \mathcal{G} , $f(v)$ will be the set of adjacent faces). Based on the properties of the cells in extension arrangements, any vertex $w \in @f(v)$ ($@f(v)$ is the boundary of $f(v)$) has a good chance to replace v (by performing $R(v, w)$) while maintaining full coverage of P . In case we find vertices on $@f(v)$ that both maintain full coverage when they replace v and shorten the corresponding route, we modify \mathcal{R} by replacing v with the one that minimizes the length of \mathcal{R} . We then replace v by the new vertex in R_g . We iterate this process and work on more cells of \mathcal{G} , as long as any improvement is achieved.

(h) We remove vertices of \mathcal{R} as we describe next. Let $u, v, w \in \mathcal{R}_g$ be defined as in step **g** above. The *removal* of v is defined as removing $\alpha(u, v)$ and $\alpha(v, w)$ from \mathcal{R} , and inserting $\alpha(u, w)$ instead.

The idea of this step is to perform removal of vertices if this operation maintains full coverage. Thus, if by replacing v , P is still covered by the modified \mathcal{R} and the rest of the routes, then we perform this removal and shorten \mathcal{R} . Let T be the vertices obtained when intersecting $\alpha(u, w)$ and \mathcal{G} . We check whether $\bigcup_{p \in T} \mathcal{V}(p)$ contains $\mathcal{V}(v)$. If so, we safely perform the removal of v . We iterate this process until no vertices can be removed.

We note that in special cases, it is possible to achieve better improvements if the coverage test is performed with the rest of the vertices in R_g , together with the vertices of T and corresponding vertices of other routes. However, this solution is very costly in terms of time and usually does not shorten the routes significantly. Thus, we do not use this idea in our implementation.

4 Implementation Details

In this section we discuss the details of the steps of our heuristics, describe the data structures that we use, and analyze the complexity of the heuristics.

(a) The time to find the static guard set using heuristic A_1 is $O(n^3)$ where n denotes the size of the input [2].

(b) Computing the visibility graph can be carried out in $O(n^2)$ time, or in $O(n \log n + k)$ time where k is the number of arcs of the visibility graph [12].

(c) We use the Floyd-Warshall algorithm to compute all pairs of the shortest paths in $O(n^3)$ time. [10]

(d) Using Prim's algorithm, we compute the minimum spanning tree in $O(n^2)$ time. [10]

(e) If the problem is $KWRP_s$, we need to remove the longest $k - 1$ edges. Finding them clearly takes $O(n \log n)$ time by sorting. If the problem is $KWRP_m$, we use parametric search for finding the minimum subtrees. We perform a binary search on the values 0 to the weight of the tree (denoted by W ; Note that $W = O(2^n)$) and stop when the interval on which we search becomes very small (we set a small constant for that). Thus, we perform $O(\log W)$ iterations. For each iteration, we do the following.

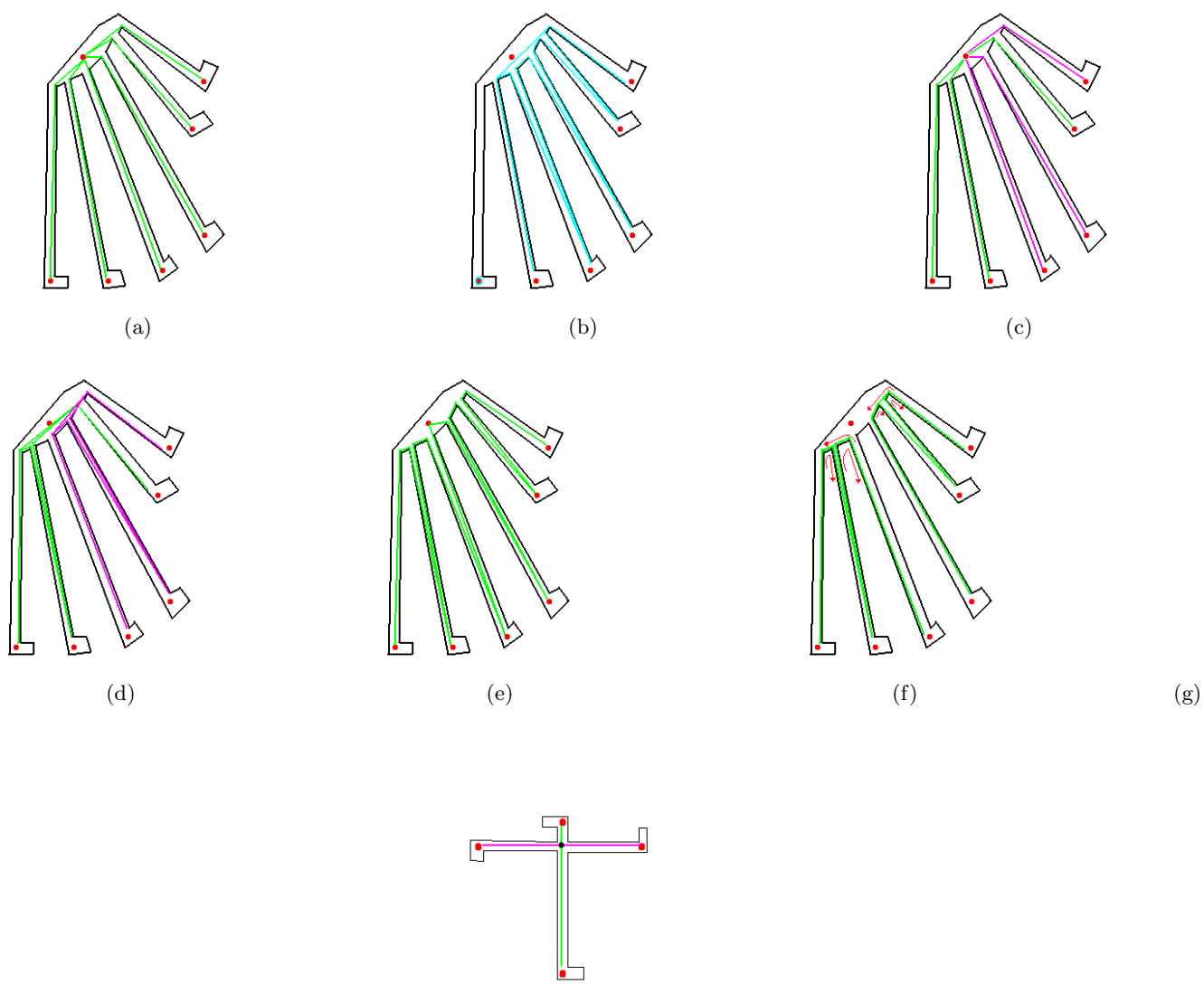


Figure 1: Improvement for step (e)

Suppose the current weight we test is W' . We visit the tree in a bottom-up fashion and remove edges once the tree below them has weight larger than W' . It is optimal for the current iteration, because if the edge that we remove is below, then it creates a smaller subtree and is clearly wasteful. Each iteration traverses the tree in $O(n)$ time. Together, this step takes $O(n \log W) = O(n^2)$ time.

(f) We use the ideas from the algorithm of Christofides in order to approximate the optimal route in $O(n^{2.5} \log^4 n)$ time.

(g) We maintain an arrangement \mathcal{A} for a union of visibility polygons and P . Note that in such an arrangement we can easily mark faces as covered or not by the visibility polygons. We initialize \mathcal{A} with the set of visibility polygons of the vertices of R_g . Then for each $v \in R_g$ we find the face that contains it, Using \mathcal{G} . We then check full coverage of P by removing and adding visibility polygons to \mathcal{A} , based on the replacement tests. We do it by checking the status of the new faces induced in \mathcal{A} (whether they are covered or not by the visibility polygons). If all new faces are covered and the corresponding route becomes shorter, we perform replacement and update \mathcal{A} accordingly. We iterate this process until no improvement is detected. Since there are $O(n)$ vertices on the routes, together they are tested for replacement with $O(n^2)$ vertices of \mathcal{A} . Since each test takes $O(n^2)$ time, the total time is $O(n^4)$.

(h) There are $O(n)$ vertices in the routes. Each can be removed at most once and each removal check requires a test with $O(n)$ visibility polygons for coverage. To carry out this test, we use an arrangement data structure similarly to step g, and use similar ideas. The induced arrangement for each vertex is thus of complexity $O(n^2)$. Thus, the total time is $O(n^3)$.

Combining all steps, we get that the time of our heuristic is $O(n^4)$. The space requirement is dominated by the resources required to maintain the geometric arrangements we use, which results in an $O(n^2)$ space.

We note that by analyzing the performance of our experiments, in all cases the asymptotic time was much smaller (bounded $O(n^3)$ or less).

5 Experiments

We have implemented our heuristics on a PC using OpenGL and CGAL (version 3.1) libraries [7]. Our software works with Microsoft Windows XP with the Visual .Net compiler. The tests were performed on a Microsoft Windows XP workstation on an Intel Pentium 4 3.2 GHz CPU with 2GB of RAM. We have performed extensive experiments with our heuristics. In this section we report our results and conclusions from our experiments. Our tests include user-generated polygons and polygons generated by a random polygon generator [4].

Since the k -watchman route problem is hard even to approximate, it is frequently difficult to evaluate the results by comparing it to optimal solutions or even solutions that approximate the optimum. Instead, we use the idea of independent witnesses (see Section 2). However, both finding the maximum independent set and finding the shortest routes that visit a set of points (for $k > 1$) are hard (the second by a simple reduction from the partition problem). In order to find independent sets, we use the software from [2] which includes a heuristic for finding independent sets inside polygons. Given such a set, we need to find an optimal solution for routes that see all the independent points (by either minimizing the longest route or minimizing the sum or route length). We made this task easier by constructing polygons manually such that the partition of independent points to watchmen was evident. Of course, this restricts the shape of the possible polygons. In the future it would be interesting to develop and implement heuristics for that

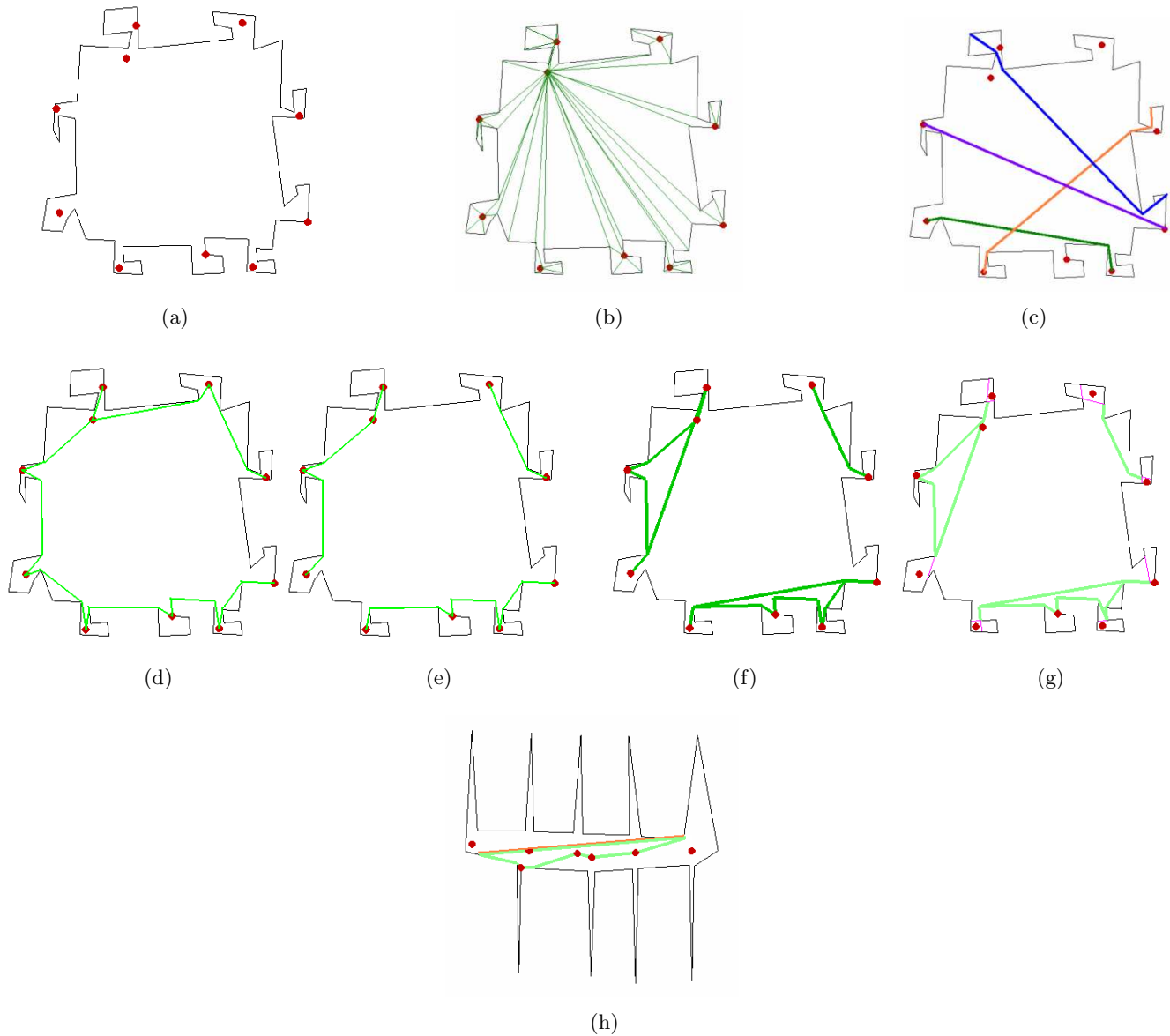


Figure 2: Illustration of $KWRP_m$. For clarity, the data in Figures (b), (c) and (g) are partially presented. The pink segments in Figure (g) are some of the extension edges that affected the output (see Section 3). In Figure (h), the green and the orange routes represent the situation before and after performing step (h), respectively.

task. We note that in many cases the lower bound measure is smaller than the optimum. Figure 10 depicts two instances, each with lower and upper bounds. Another example of lower bound is demonstrated in Figure 6(b), where the route can be obtained as a lower bound if we take two independent points inside the left and right ‘pockets’ (compare to Figure 6(a)).

We note that our heuristics were always within a factor of 5 from the lower bound. Most were even within a factor of 2 and many were within a factor of 1.5. We want to emphasize that in many cases the lower bound routes did not cover the polygons, thus were not tight. It is worth noting that the polygons with highest factor were the ones whose vertex visibility graph was large, and that have a set of guards that are located in a small neighborhood and for which step (a) produced a set of guards that are not close (a good example for this type is in Figure 6(a) and 6(b)). However, we note that since it is impossible to quantify the type of the polygons and define the ‘average’ polygon, and also because this experiment is constrained to a limited set of polygons, we cannot make a precise quantification for this experiment. Nevertheless, we believe that being a factor of 5 from the lower bounds in our tests shows some kind of a *constant-factor approximation* behavior for various types of polygons.

Even in cases where we could not find lower bounds easily, we were usually satisfied with the results obtained with our software on many types of polygons. In many cases our results seemed to be very close to optimal by intuition.

Figure 7 shows the routes obtained by our software for different kinds of polygons with different values of k .

There are many parameters that seem to affect the time taken to run our software. The main ones that can be quantified are the size of the polygons (including holes), the number of watchmen and the size of \mathcal{G} . As for the latter, in polygons with wide areas (like the example in Figure 2), \mathcal{G} is large because many extension edges intersect. On the other hand, polygons with narrow passages (as the common randomly-generated polygon), result in smaller sizes of \mathcal{G} . We note that our experiments showed that the selection of k (number of mobile watchmen) did not have any significant effect on the time. Running times were always very close, and moreover, it seems that there is no correlation between k and the run time. In our results the time refers to an average of four runs, with $k = \{1, 2, 3, 4\}$. In Figures 3,4 and 5 we plot the time as a function of these parameters. Figure 3 shows the results of many kinds of polygons. Other than that, we devote separate graphs for random polygons (Figures 4 and 7(i)) and spike box polygons (Figures 5 and 7(b)), where different size of polygons are tested. These two classes of polygons are different in the following two senses. In the spike box polygons \mathcal{G} is large while in the random polygon class \mathcal{G} is small. Another difference is that the size of independent sets in spike box polygons is usually much smaller than in random polygons which tend to have narrow passages. For these reasons, we choose these two distinct classes for experiments that show the behavior as a function of the polygon’s size. In each figure there are two graphs, for $KWRP_m$ and $KWRP_s$. The graphs show that there is a correlation between the time and both the polygon’s size and the number of guards (although there are exceptions). On the other hand, we could not find any correlation between the size of \mathcal{G} and the time. Finally, we note that the times for $KWRP_m$ and $KWRP_s$ were usually very similar and there is no evidence for either one to be faster than the other in any set of polygons.

6 Directions for Future Improvements

The main effort in our heuristics is to construct short routes. We first construct routes (steps a-f) and then try to shorten them (steps g-h). In most polygons we tested, all steps were useful for constructing short routes and in the majority of the tests, steps g and h improved the results. However, we also found some

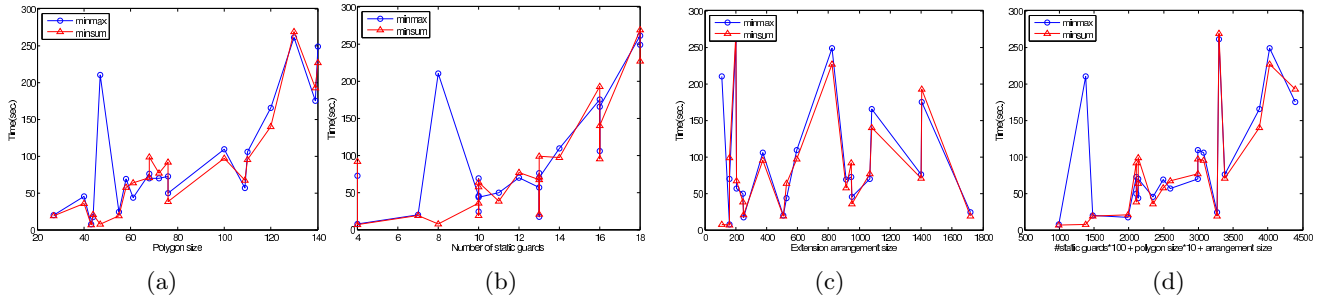


Figure 3: Time as a function of different parameters for different polygons

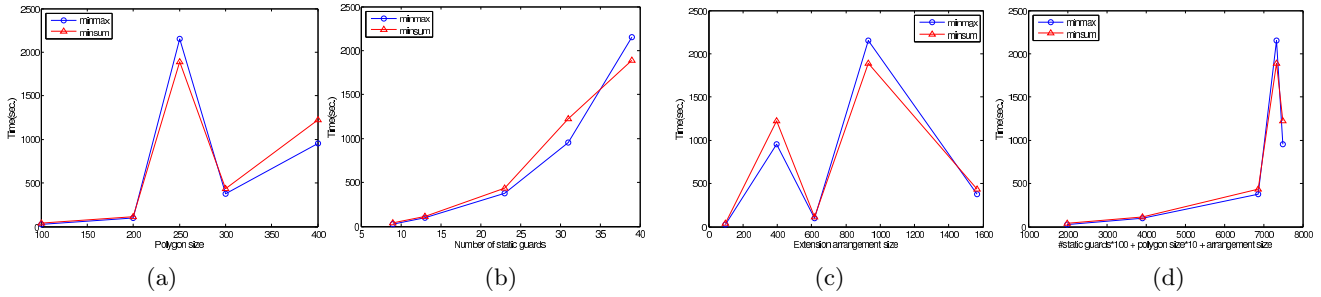


Figure 4: Time as a function of different parameters for randomly generated polygons

polygons for which further special improvements would improve the results. In this section we describe these ideas. We first emphasize that these ideas will be useful on only the minority of the instances we tested, and in these cases it does not seem that they will improve the results by much. Nevertheless, we feel that it is useful to present them here. We also note that as for today, we only have ideas on what the improvement should do. These ideas seem to require solutions that are NP-hard, very challenging, do not have clear heuristics, and seem to take much processing time. Based on these observations, we decided not to consider them for implementation in this work. However, they can motivate the study of developing heuristics that will both improve the results and be time-efficient.

One of the ideas was described in Section 3 (Steiner points and splitting around vertices). We described a restricted method for improvement, one that even motivates more work for improvement on its own. However, as mentioned, generally applying all of these ideas seems like a very challenging future direction. We next present three more ideas and demonstrate how they can improve the results.

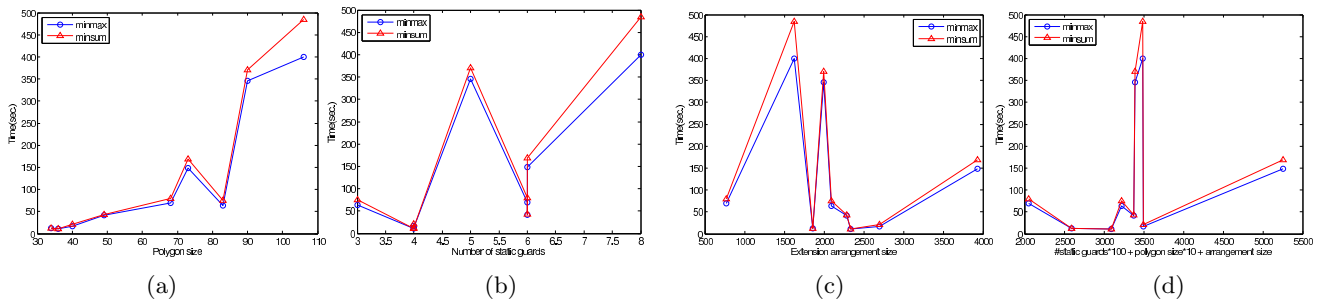


Figure 5: Time as a function of different parameters for spike box polygons

- In step **a** we obtain a list of static guards that cover the polygon. Intuitively, the closer this set is, the shorter the potential routes become. However, distances between guards were not considered for guarding in [2]. Moreover, it is possible that a non-optimal set will give better results for the watchman problem. The idea here is to find a set of static guards that cover the polygon, by considering the distance measure between them. One simple idea to implement is the following. Since the algorithm of [2] is incremental by selecting the guards, we could break ties based on the diameter obtained with the current set of guards. However, we note that it would clearly be only a partial solution. Consider Figures 6(a) and (b). Figure 6(a) is the result we obtained. While in Figure 6(b) the set of static guards (note that it covers the polygon) has a smaller diameter, it contains more guards. This set of close guards gives the route depicted in Figure 6(b) which is 60% shorter than the one obtained with our implementation.
- In step **d** we build the minimum spanning tree \mathcal{T} of the static guards. We note that local changes to \mathcal{T} may improve the results (but we emphasize that this is usually not the case). Consider the polygon in Figures 6(c)-(f). \mathcal{T} and the results obtained with our software are shown in Figures 6(c) and (d), respectively. In Figure 6(e) another spanning tree is depicted (it is obtained by a local change on \mathcal{T}). The results with the new tree are better as shown in Figure 6(f). The improvement is achieved by not moving along one of the edges twice (it is surrounded in Figures 6(d) and (f)). In this case the improvement is roughly 6%.
- In steps **g** and **h**, we use only local modifications. However, in special cases it is possible that modifications that involve places on the routes that are not local could help. In step **g** the idea is to check more vertices of \mathcal{G} for shortening the routes, and in step **h** the idea is to check whether far places on the routes can replace a certain guarding vertices. We note that although they would be easy to adopt, the extra processing time they require is very significant if we simply change our implementation and not devise a clever way to perform them efficiently.

7 Conclusions and future work

We presented heuristics for constructing k -watchman routes inside polygons, possibly with holes. As far as we know, this is the first attempt to develop heuristics for this problem. We implemented these heuristics and conducted experiments. We tested our software with many polygons and presented our results in detail. In limited cases we were even able to evaluate our results by comparing with lower bounds, and obtained a bound of factor-5 approximation in our tests. Moreover, many other results in many kinds of polygons look efficient and not far from optimal solution by intuition.

We proposed several directions for improving the quality of our results in Section 6. It would also be desirable to find ways to improve the time bounds of the different steps in our heuristics. Additionally, it would be interesting to explore ideas for practical implementations of lower bounds. Another direction is to develop heuristics for other kinds of watchman problems such as the ones mentioned in Section 1. Finally, we note that it would be desirable to find a way to prove efficient lower bounds to our heuristic and to develop approximation algorithms for restricted versions.

References

- [1] M. H. Alsuwaiyel and D. T. Lee. Finding an approximate minimum-link visibility path inside a simple polygon. *Inf. Proc. Lett.*, 55(2):75–59, 1995.

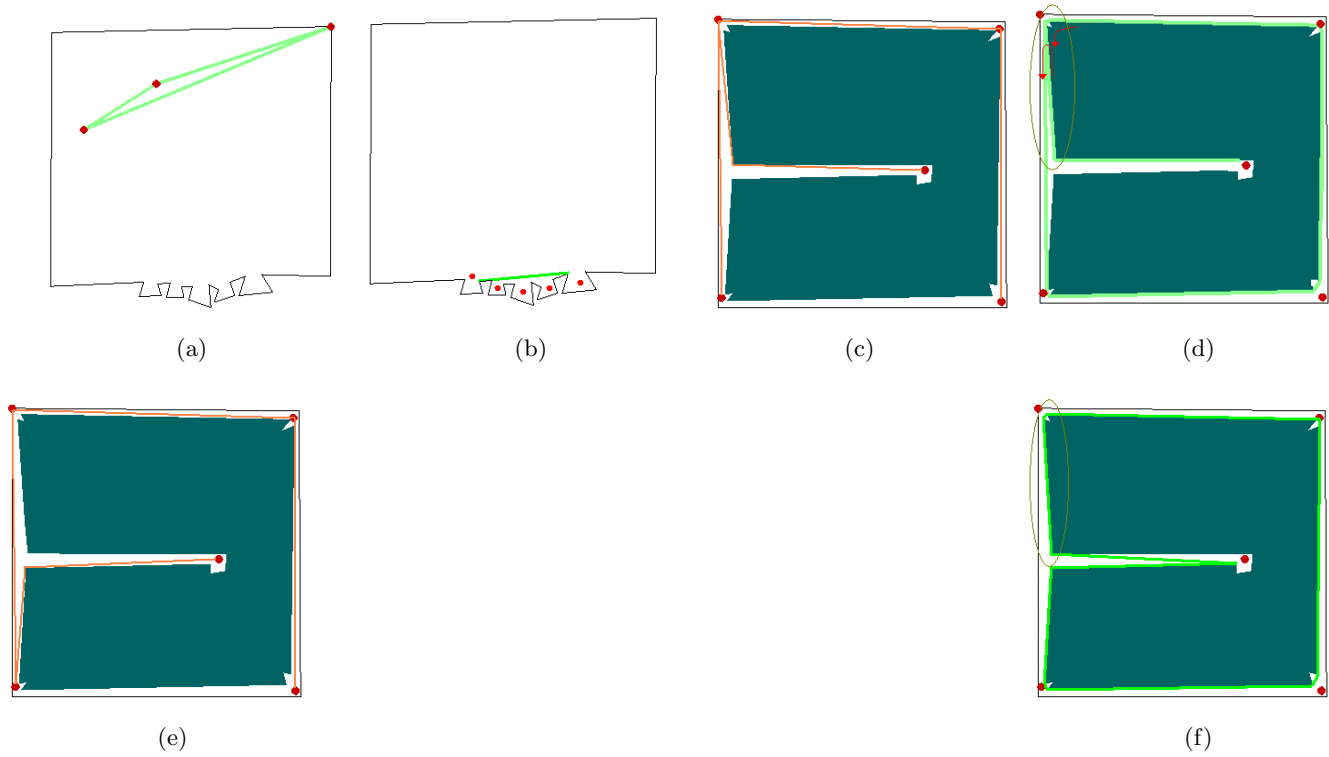


Figure 6: Ideas for future improvements. In this figure and others that follow, holes are painted in dark green.

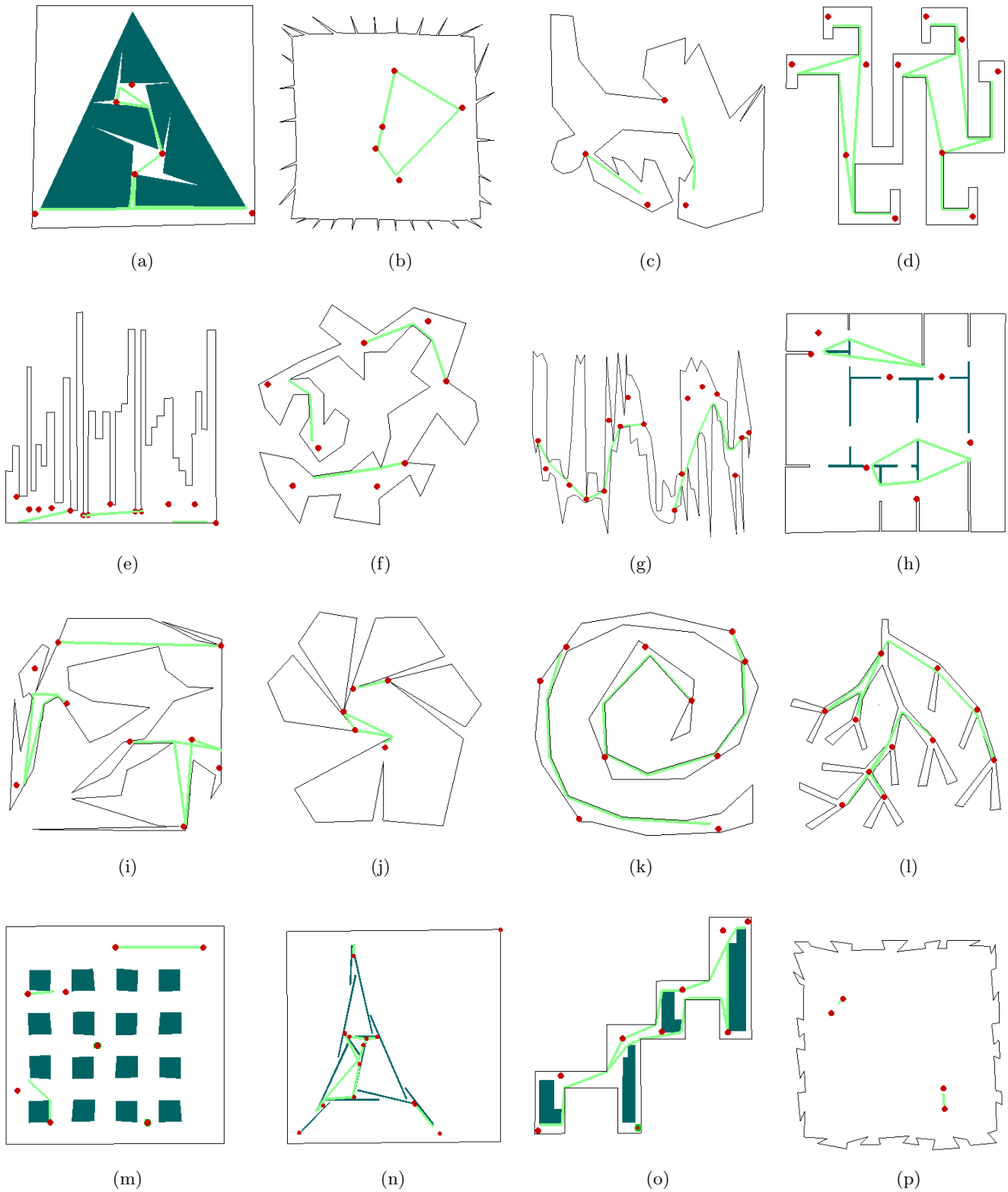


Figure 7: Experiment snapshots obtained with our software on different kinds of polygons. Subfigures (a) and (b) show results for $k = 1$. Subfigures (c)-(l) shows results of $KWRP_m$ while subfigures (m)-(p) shows results of $KWRP_s$. For convenience, we marked the static guards from step (a) in red discs. Mobile Watchmen that have 0-length route are depicted with green discs.

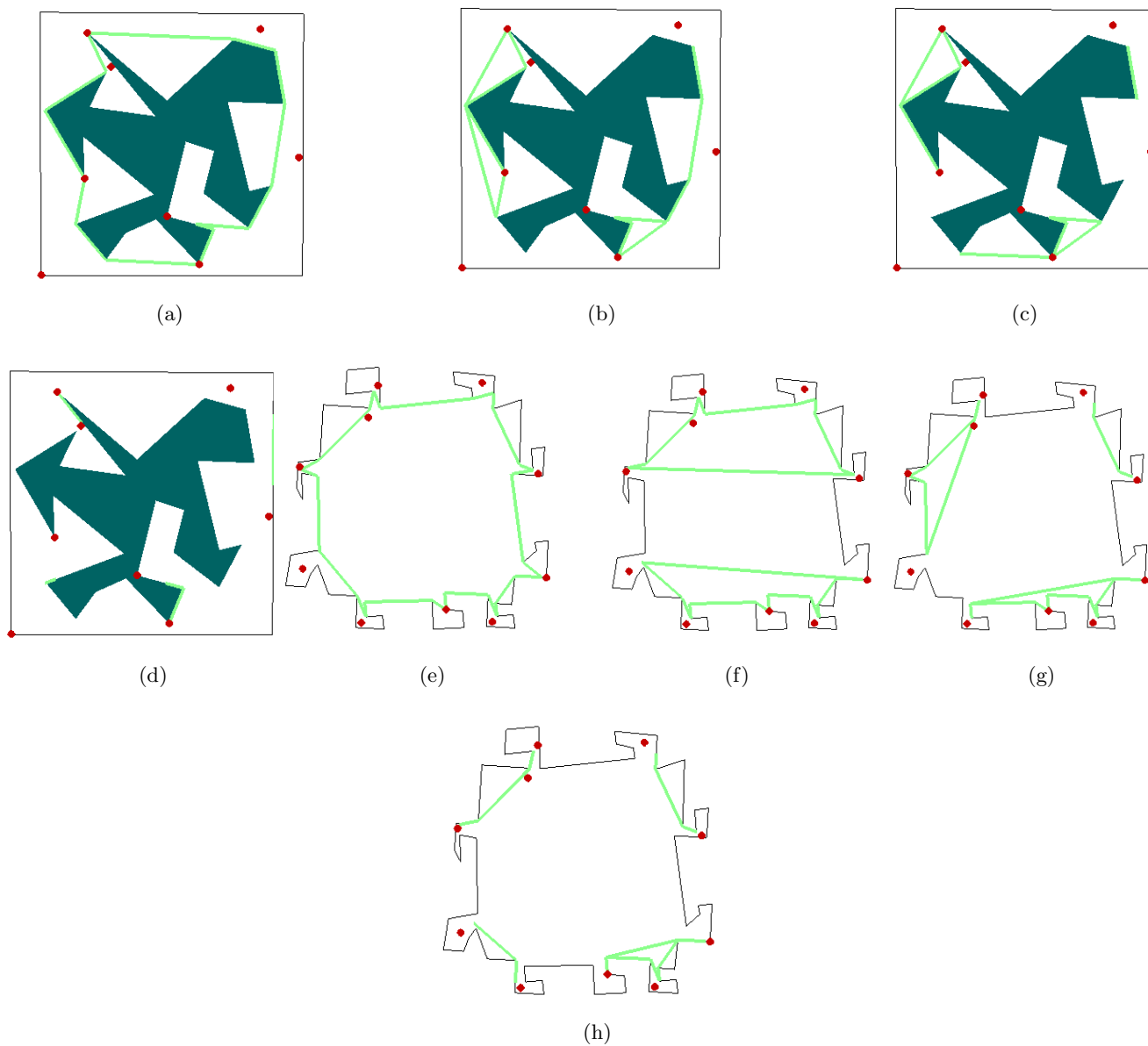


Figure 8: Experiments showing the results for different number of watchmen.

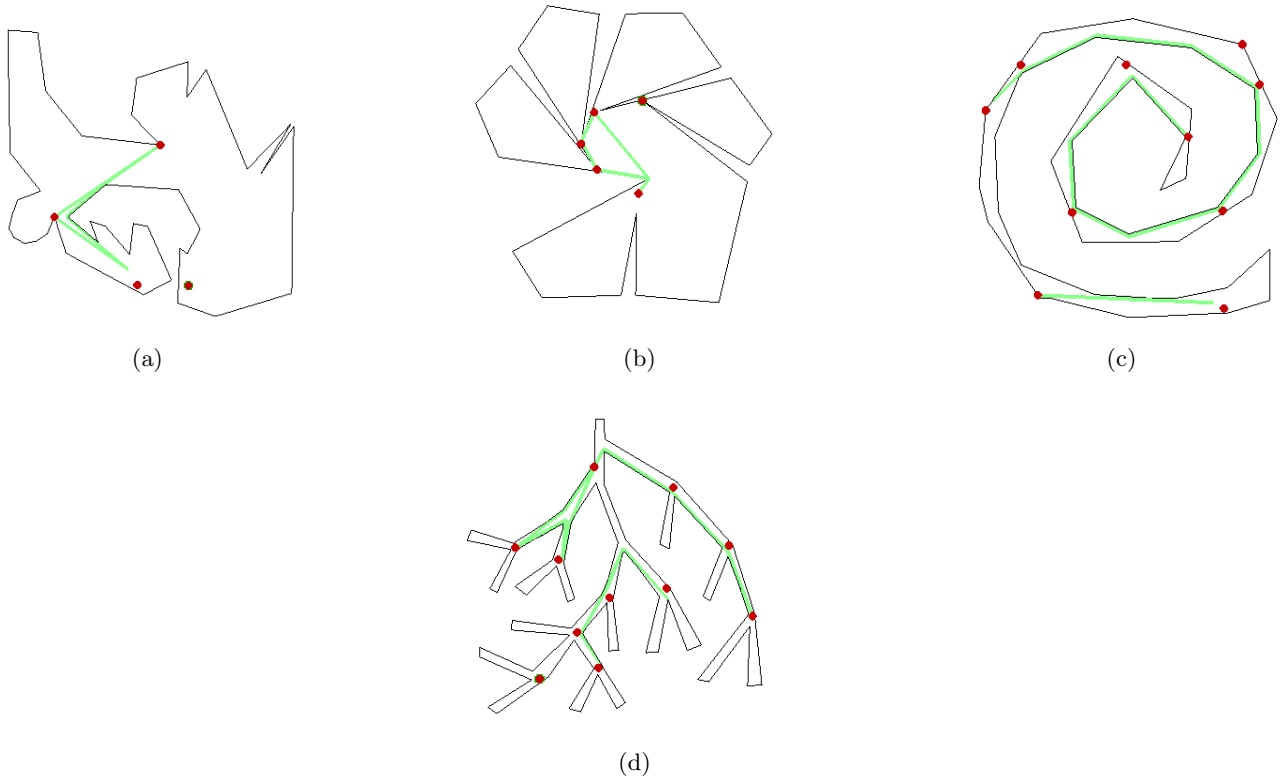


Figure 9: Some results obtained with $KWRP_s$ (Compare with the results of $KWRP_m$ on the same polygons in Figure 7).

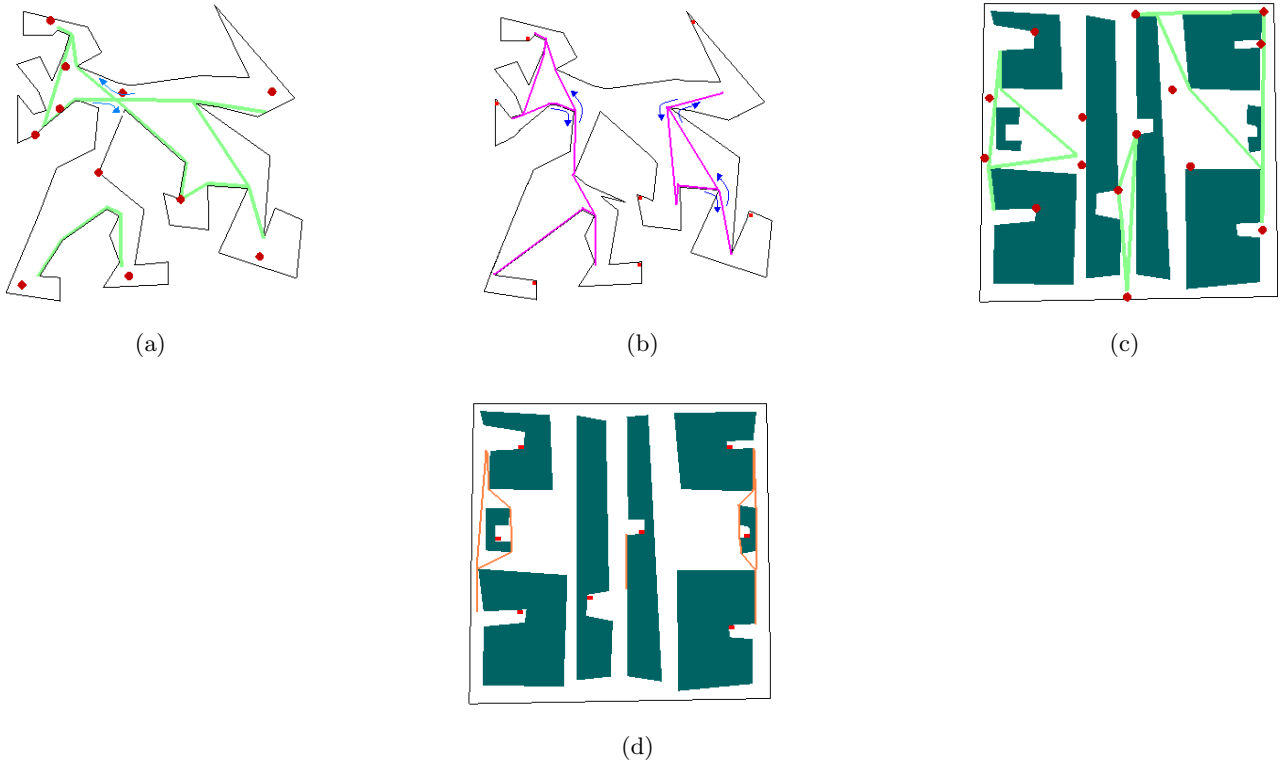


Figure 10: Comparing the results with lower bounds. Subfigures (a) and (c) depict upper bounds obtained with our implementation, while Subfigures (b) and (d) depicts corresponding lower bounds where the red squares represent independent witness points. Note that the longest route in (a) self intersects. In such cases, it is easy to eliminate crossings as depicted in the figure.

- [2] Y. Amit, J. Mitchell, and E. Packer. Locating guards for visibility coverage of polygons. *Workshop on Algorithm Engineering and Experiments, ALANEX*, 2007.
- [3] E. Arkin, J. Mitchell, and C. Piatko. Minimum-link watchman tours. *Technical Report, University at Stony Brook*, 1994.
- [4] T. Auer and M. Held. Heuristics for the generation of random polygons. *Proc. 8th Canad. Conf. Computat. Geometry*, 1996.
- [5] Carlsson, Jonsson, and Nilsson. Finding the shortest watchman route in a simple polygon. In *ISAAC: 4th International Symposium on Algorithms and Computation (formerly SIGAL International Symposium on Algorithms)*, Organized by Special Interest Group on Algorithms (SIGAL) of the Information Processing Society of Japan (IPSI) and the Technical Group on Theoretical Foundation of Computing of the Institute of Electronics, Information and Communication Engineers (IEICE), 1993.
- [6] S. Carlsson, B. J. Nilsson, and S. C. Ntafos. Optimum guard covers and m-watchmen routes for restricted polygons. In *Workshop on Algorithms and Data Structures*, pages 367–378, 1991.
- [7] *The CGAL User Manual, Version 3.1*, 2004. www.cgal.org.
- [8] W. Chin and S. Ntafos. Optimum watchman routes. *Inform. Process. Lett.*, 1988.
- [9] N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. *Report 388, Graduate School of Industrial Administration, CMU*, 1976.
- [10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, September 2001.
- [11] L. Gewali and S. C. Ntafos. Watchman routes in the presence of a pair of convex polygons. *Information Sciences*, 105(1-4):123–149, 1998.
- [12] S. K. Ghosh and D. M. Mount. An output-sensitive algorithm for computing visibility graphs. In *SIAM J. computing*, pages 888–910, 1991.
- [13] C. Icking and R. Klein. The two guards problem. *International Journal of Computational Geometry and Applications*, 2(3):257–285, 1992.
- [14] D. T. Lee and A. K. Lin. Computational complexity of art gallery problems. *IEEE Trans. Info. Th IT-32*, pages 276–282, 1986.
- [15] J. Mitchell and E. Wynters. Watchman routes for multiple guards. *Proc. 3th Canad. Conf. Computat. Geometry*, pages 126–129, 1991.
- [16] B. Nilsson. Guarding art galleries - methods for mobile guards. *PhD thesis, Lund University*, 1995.
- [17] J. O’Rourke. *Art gallery theorems and algorithms*. Oxford University Press, Oxford, 1987.
- [18] S.-M. Park, J.-H. Lee, and K.-Y. Chwa. Visibility-based pursuit-evasion in a polygonal region by a searcher. *Lecture Notes in Computer Science*, 2076, 2001.
- [19] T. Shermer. Recent results in art galleries. *Proc. of the IEEE*, pages 1384–1399, 1992.
- [20] X. Tan. Fast computation of shortest watchman routes in simple polygons. *Inf. Proc. Lett.*, 77:27–33, 2001.
- [21] X. Tan, T. Hirata, and Y. Inagaki. An incremental algorithm for constructing shortest watchman routes. *Int. J. Comput. Geometry Appl. (IJCGA)*, 3(4):351–365, 1993.
- [22] J. Urrutia. Art gallery and illumination problems. In J. Sac and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 973–1027. Elsevier Science Publishers, Amsterdam, 2000.
- [23] M. Yamashita, H. Umemoto, I. Suzuki, and T. Kameda. Searching for mobile intruders in a polygonal region by a group of mobile searchers (extended abstract). In *Symposium on Computational Geometry*, pages 448–450, 1997.