A COMPLETE ALGORITHM FOR SEARCHLIGHT SCHEDULING

KARL J. OBERMEYER ANURAG GANGULI FRANCESCO BULLO

Center for Control, Dynamical Systems, and Computation,
University of California at Santa Barbara,
Santa Barbara, CA 93106, USA
karl.obermeyer@gmail.com, anurag.ganguli@gmail.com, bullo@engineering.ucsb.edu

Received Oct. 2008
Revised Oct. 2010
Communicated by Kokichi Sugihara

This article develops an algorithm for a group of guards statically positioned in a nonconvex polygonal environment with holes. Each guard possesses a single *searchlight*, a ray sensor which can rotate about the guard's position but cannot penetrate the boundary of the environment. A point is detected by a searchlight if and only if the point is on the ray at some instant. Targets are points which move arbitrarily fast. The objective of the proposed algorithm is to compute a schedule to rotate a set of searchlights in such a way that any target in an environment will necessarily be detected in finite time. This is known as the *Searchlight Scheduling Problem* and was described originally in 1990 by Sugihara et al. We take an approach known as exact cell decomposition in the motion planning literature. The algorithm operates by decomposing the searchlights' joint configuration space and the environment, and then by constructing a so-called information graph. Searching the information graph for a path between desired states yields a search schedule. We also introduce a new problem called the *ϕ-Searchlight Scheduling Problem* in which *ϕ-searchlights* sense not just along a ray, but over a finite field of view. We show that our results for searchlight scheduling can be directly extended for ϕ-searchlight scheduling. Proofs of completeness, complexity bounds, and computed examples are presented.

Keywords: searchlight scheduling; visibility; pursuit-evasion; computational geometry; exact cell decomposition; motion planning.

1. Introduction

Consider a group of point guards statically positioned in a nonconvex polygonal environment with holes, e.g., a floor plan. Each guard is equipped with a single *searchlight*, a ray sensor which can rotate about the guard's position but cannot penetrate the boundary of the environment (imagine a ray of light such as a laser range finder, or a camera with a very narrow field of view). A searchlight aims only in one direction at a time and cannot penetrate the boundary of the environment, but its direction can change continuously. A point is detected by a searchlight at some instant if and only if the point lies on the ray. Targets are points which move arbitrarily fast. The *Searchlight Scheduling Problem* is to

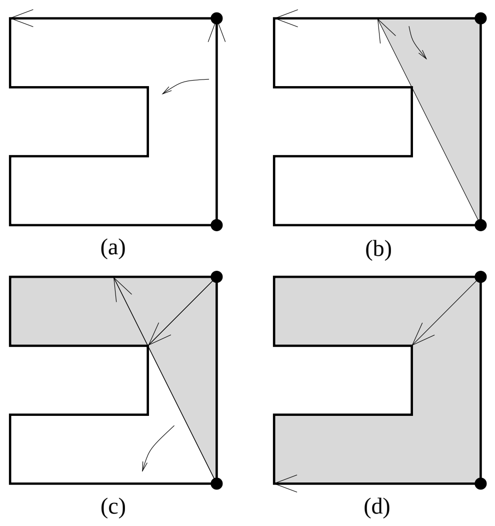

Fig. 1. A simple example of a searchlight schedule. “Clear” regions where no undetected evader could exist are shown in gray. From (a) to (d): First the lower searchlight aims at the upper searchlight and sweeps until it hits a corner where its visibility is occluded. Next, the upper searchlight sweeps the area the lower searchlight cannot see. Finally, the lower searchlight continues sweeping the remainder of the environment. No target, no matter how fast, would be able to avoid detection by this rotation sequence.

> Find a schedule to rotate a set of stationary searchlights such that any target in an environment will necessarily be detected in finite time.

A searchlight problem instance consists of an environment together with a set of stationary sensor positions. Obviously there can only exist a search schedule if all points in the environment are visible from some guard. For a graphical description of our objective see Fig. 1.

To our knowledge the Searchlight Scheduling Problem was first introduced by Sugihara, Suzuki and Yamashita.[1] They give a solution, the “One Way Sweep Strategy”, to the limited class of searchlight scheduling problem instances in which the environment is simply connected and there is at least one searchlight located on the boundary for every connected component of their visibility graph. In Ref. 2, a linear time algorithm is given for finding searchlight schedules in the special case of polygons with searchlights on the boundary and no holes. Ref. 3 gives upper bounds on the number of guards with multiple searchlights sufficient in polygonal environments containing holes. We adopt the convention in Ref. 3 and call a mobile guard possessing k searchlights a *k-searcher*. Some articles involving 1-searchers, sometimes calling them *flashlights* or *beam detectors*, are Ref. 4, 5, 6, and 7. Closely related is the Classical Art Gallery Problem, namely that of finding a minimum set of guards (with omnidirectional vision) such that the entire polygon is visible. There are many variations on the art gallery problem which are wonderfully surveyed in Ref. 8, 9, and 10. With an emphasis on practical imaging considerations, Ref. 11 describes a centralized task-specific procedure for choosing the locations of cameras in a network. As described in Ref. 12 and 13, there are distributed algorithms for deploying guards into simple polygons such that their final positions are a solution to the art gallery problem. We later used these ideas in Ref. 14 for distributed deployment of agents such that the agents are able to execute a searchlight schedule in an asynchronous distributed manner from their final positions.

Exact cell decomposition, a method we use in the present article, has been used in the design of complete algorithms to solve visibility-based pursuit-evasion problems before, e.g., in Ref. 15 and 4. Ref. 15 gives an algorithm for a single mobile searcher with omnidirectional vision, and it is shown that determining the minimum number of such pursuers required to clear a polygonal environment with holes is NP-hard. Ref. 4 describes a complete algorithm for a single mobile "ϕ-searcher" having an angle ϕ field of view, and it is shown that determining the minimum number of such pursuers required to clear a polygonal environment with holes is also NP-hard. Due to anticipated computational complexity, both articles, perhaps appropriately, dismiss the idea of using a complete exact cell decomposition for the case of multiple searchers, although they do implement incomplete extensions which they claim work well for practical purposes. It is suggested on page 569 of Ref. 16 that implementation of a complete exact cell decomposition algorithm for multiple pursuers would be further complicated because "some of the cell boundaries are algebraic surfaces due to complicated interactions between the visibility polygons of different pursuers." To our knowledge nobody has carried out the design of a complete algorithm to solve any visibility-based pursuit-evasion problem involving arbitrary polygonal environments with holes. However, there are at least two noteworthy articles involving multiple pursuers in polygonal environments without holes. Ref. 17 provides a polynomial time complete algorithm for two 1-searchers in a simple polygonal environment, but has not been extended to three or greater pursuers and it is not clear how to do so. Ref. 18 gives a polynomial time complete algorithm to determine the minimum number of ∞-searchers (omnidirectional vision) necessary to clear a simple polygon, but under the constraints that (1) the pursuers are in a chain configuration where consecutive pursuers along the chain are mutually visible, and (2) end pursuers must remain on the polygon boundary.

There are three main contributions in this article. First, we show by exact cell decomposition that if an instance of the Searchlight Scheduling Problem permits any solution at all, then it also permits a solution in a reduced discrete solution space. The second contribution is to use the knowledge of the solution space discretization to design a complete[a] algorithm for searchlight scheduling. Although it remains an open problem whether searchlight scheduling is NP-hard, our computed

[a]Here *complete* means that if a solution exists, the algorithm is guaranteed to find one in finite time.

examples demonstrate that for searchlights, even in environments with holes, the time complexity of a complete exact cell decomposition is not entirely prohibitive and can be practical for problem instances of useful size. To accomplish this, we construct our cell decomposition dependent on the pursuer positions so that there is no need to explicitly compute any algebraic surfaces. At this time no other algorithm exists to solve the general Searchlight Scheduling Problem. As a third contribution we treat a new problem which we call the *ϕ-Searchlight Scheduling Problem* in which *ϕ-searchlights* sense not just along a ray, but over a finite field of view (see Fig 13). We show how our searchlight scheduling algorithm can be extended to take advantage of ϕ-searchlights having a wider field of view than just a ray. This is an important extension because for cameras having a finite field of view it is a much more realistic sensor model. We envision our algorithms and/or other algorithms inspired by this work will one day be used in automating the design of security systems consisting of networks of statically positioned rotating sensors and actuators.

This article is organized as follows. Section 2 covers preliminary notation, technical definitions, and statement of assumptions. Section 3 provides an extensive theoretical development which culminates in a proof showing a reduction of the searchlight scheduling solution space. The solution space reduction guarantees the completeness of our algorithm, which we present in Section 4. In Section 5 we introduce the ϕ-Searchlight Scheduling Problem and explain how our results for searchlight scheduling can be directly extended for ϕ-searchlights. We conclude in Section 6.

2. Preliminaries

2.1. *Notation*

We begin by introducing some basic notation. We let $\mathbb{R}$ and $\mathbb{T}^d$ represent the set of real numbers and the d-dimensional torus, respectively. Clockwise is abbreviated by cw, and counterclockwise by ccw. Given two points $a, b \in \mathbb{R}^2$, we let $[a, b]$ signify the *closed segment* between a and b. Similarly, $]a, b[$ is the *open segment* between a and b, $[a, b[$ represents the set $]a, b[\cup \{a\}$ and $]a, b]$ is the set $]a, b[\cup \{b\}$. Given a set (resp. list) A, $|A|$ denotes the cardinality of the set (resp. list), A° the interior, $\bar{A}$ the closure, and ∂A the boundary. Also, we shall use P to refer to tuples of elements in $\mathbb{R}^2$ of the form $(p^{[0]}, \ldots, p^{[N-1]})$ (these will be the locations of the searchlights), where N denotes the total number of searchlights.

We turn our attention to the environments we are interested in and to the concepts of visibility in such environments. The environment, denoted by $\mathcal{E}$, is closed and consists of an (outer) polygon containing polygonal holes. The boundaries of these polygons do not intersect themselves or each other. Throughout this article, n will refer to the number of edges of $\mathcal{E}$ (holes included) and r the number of reflex vertices. A reflex vertex is constituted by any concave vertex of the outer polygon or convex vertex of a hole. A point $q \in \mathcal{E}$ is *visible from* $p \in \mathcal{E}$ if $[p, q] \subset \mathcal{E}$. The *visibility set* $\mathcal{V}(p) \subset \mathcal{E}$ from a point $p \in \mathcal{E}$ is the set of points in $\mathcal{E}$ visible from p. A *visibility gap* of a point $p \in \mathcal{E}$ is defined as any line segment $[a, b]$ such that $]a, b[\subset \mathcal{E}^\circ$, $[a, b] \subset \partial\mathcal{V}(p)$, and it is maximal in the sense that $a, b \in \partial\mathcal{E}$. Intuitively, visibility gaps form the border between portions of $\mathcal{E}$ visible from p and portions of $\mathcal{E}$ not visible from p.

Now we introduce some notation and definitions specific to the Searchlight Scheduling Problem. An instance of the Searchlight Scheduling Problem is specified by a pair $(\mathcal{E}, P)$, where $\mathcal{E}$ is an environment and P is a set of searchlight locations in $\mathcal{E}$. For convenience, we will refer to the ith searchlight as $s^{[i]}$ (which is located at $p^{[i]} \in \mathbb{R}^2$), and $S = \{s^{[0]}, \ldots, s^{[N-1]}\}$ will be the set of all searchlights. We let $\theta^{[i]}$ denote the configuration angle of the searchlight in radians from the positive horizontal axis, and $\Theta = \{\theta^{[0]}, \ldots, \theta^{[N-1]}\}$ denote the joint configuration. So, if we say, e.g., aim $s^{[i]}$ at point e, what we really mean is set $\theta^{[i]}$ equal to an angle such that the ith searchlight is aimed at e. Note that searchlights do not block visibility of other searchlights.

The next few definitions are similar to those in Ref. 1.

Definition 1 (Schedule). Let $[0, T]$ be a finite interval of real time. A *schedule* of a searchlight $\theta^{[i]}(t) \in \Theta(t)$ is a continuous function $\theta^{[i]} : [0, T] \to \mathbb{T}^1$ such that $s^{[i]}$ changes direction of rotation at most a finite number of times.

The requirement that no searchlight switches direction of rotation an infinite number of times is important for practical realizability.

Definition 2 (Active). Searchlight $s^{[i]}$ is *active* at time t if it is rotating (has nonzero angular velocity), otherwise it is *inactive*.

The *ray* of $s^{[i]}$ at time $t \in [0, T]$ is the intersection of $\mathcal{V}(p^{[i]})$ and the semi-infinite ray starting at $p^{[i]}$ with direction $\theta^{[i]}(t)$. Searchlight $s^{[i]}$ is said to be *aimed* at a point $e \in \mathcal{E}$ at some time instant if e is on the ray of $s^{[i]}$. A point e is *illuminated* if there exists a searchlight aimed at e.

Definition 3 (Separability). Two points in $\mathcal{E}$ are *separable* at time $t \in [0, T]$ if every continuous path connecting them in the interior of $\mathcal{E}$ contains an illuminated point, otherwise they are *nonseparable*. Two regions R_1 and R_2 in $\mathcal{E}$ are *separable* if every pair of points $e_1 \in R_1$ and $e_2 \in R_2$ are separable.

Definition 4 (Contamination and Clarity). A point $e \in \mathcal{E}$ is *contaminated* at time zero if and only if it is not illuminated. The point e is contaminated at time $t \in]0, T]$ if and only if there exists a continuous function $f : [0, t] \to \mathcal{E}$ such that $f(t) = e$ and for every instant $t' \in [0, t]$, $f(t')$ is not illuminated by any searchlight. A point which is not contaminated is called *clear*. A region is said to be *contaminated* if it contains a contaminated point, otherwise it is *clear*.

Definition 5 (Search Schedule). Given $\mathcal{E}$ and a set of searchlight locations $P = \{p^{[0]}, \ldots, p^{[N-1]}\}$, a schedule $\Theta(t) = \{\theta^{[0]}(t), \ldots, \theta^{[N-1]}(t)\} : [0, T] \to \mathbb{T}^N$ is a *search schedule for* $(\mathcal{E}, P)$ if $\mathcal{E}$ is clear at T.

2.2. *Assumptions*

The following *main assumptions* will be made about every problem instance in this article:

(i) The environment is static and has a finite number of vertices.

(ii) Every point in the environment is visible from some searchlight and there is a finite number N of searchlights.
Comments: If there were some point in the environment not visible from any searchlight, then a target could remain there undetected for all time.

(iii) No two searchlights are co-located.

(iv) All searchlights are switched on at all times, even when rotating.
Comments: Leaving inactive searchlights switched on can only increase, and not decrease the chance of detecting an evader. One might argue that leaving a sensor switched on could be costly, e.g., from an energy standpoint, but we are not considering such things in our solution.

3. Reducing the Solution Space

The solution space of the Searchlight Scheduling Problem is the set of all possible schedules. This section focuses on defining several special classes of schedules and showing that the existence of a search schedule in the most general continuous class implies the existence of a search schedule in a reduced discrete class which can be searched for a solution. This is accomplished by an exact cell decomposition of the searchlights' joint configuration space ($\mathbb{T}^N$). Our algorithm in Section 4 and its completeness will follow directly from the solution space reduction.

At any instant of a schedule, searchlights are aimed in various directions so that their beams separate (in the sense of Definition 3) the environment $\mathcal{E}$ into a set of distinct polygonal regions (possibly containing holes) each of which is either entirely clear or entirely contaminated. We formalize this.

Definition 6 (Maximal Nonseparable Region). For a fixed searchlight configuration Θ and an unilluminated point $e \in \mathcal{E}$, the equivalence class of all points in $\mathcal{E}$ which are nonseparable from e is called a *maximal nonseparable region.*

At any time during a schedule there is a finite number of maximal nonseparable regions. As an example, in Fig. 1(c) there are 4 maximal nonseparable regions, 3 of which are clear.

Definition 7 (Support). If a portion of a searchlight's beam forms part of the boundary of a maximal nonseparable region, then that searchlight is said to *support* that maximal nonseparable region. The set of all searchlights whose beams form the boundary of a maximal nonseparable region is called the *support* of that maximal nonseparable region.

The support of a maximal nonseparable region changes, in general, over the course of a schedule. As a schedule is executed, maximal nonseparable regions, in addition to continuously deforming, may undergo any of the following changes:

Disappear: A maximal nonseparable region disappears if and only if its area goes to zero. This can happen if one or more searchlights rotate (i) into $\partial\mathcal{E}$, (ii) into coincidence with another searchlight's beam, (iii) onto the intersection of other searchlights' beams, or (iv) onto the intersection point of another searchlight's beam with $\partial\mathcal{E}$. The only way to clear a contaminated maximal nonseparable region is to make it disappear. Examples are shown in Fig 2.

Appear: The reverse of disappear. Note any maximal nonseparable region which appears remains clear until merging with a contaminated region.

Merge: Two or more maximal nonseparable regions can merge into one if a searchlight rotates (i) past a reflex vertex of $\partial\mathcal{E}$ where its visibility is occluded, (ii) away from a reflex vertex which it was grazing, or (iii) past another searchlight not on $\partial\mathcal{E}$. Examples are shown in Fig. 3. A clear maximal nonseparable region can become contaminated only by merging with a contaminated region.

Split: The reverse of merge.

Indeed, any possible change to a maximal nonseparable region will fall under one of the above categories. To describe the entire system evolution precisely, we say at time t it possesses an *information state* which consists of the searchlights' joint configuration $\Theta(t) \in \mathbb{T}^N$ together with the *contamination state* $C(t)$ of the environment $\mathcal{E}$. By contamination state is meant an encoding of which points in $\mathcal{E}$ are contaminated, e.g., a binary labeling of the maximal nonseparable regions (0 for clear, 1 for contaminated). We denote the information state by a pair $(\Theta(t), C(t))$, or by (Θ, C) when the time is implicitly understood. Note, however, that the concept of information state has no intrinsic dependence on time, so we may also speak of the information state of a searchlight system without it being associated with any particular schedule. The information state takes on a value in a continuous information space $\mathcal{I}$. To every searchlight schedule corresponds a unique trajectory through $\mathcal{I}$, thus a search schedule can simply be viewed as an information space trajectory which begins with a completely contaminated information state and ends with a completely clear information state. Ultimately we will discretize the continuous information space $\mathcal{I}$ into a so-called *information graph* $\mathcal{G}_{\mathcal{I}}$ which can be searched systematically for a searchlight rotation schedule. We delay discussing the information graph further until Section 4.

The definitions to follow will allow us to describe the exact cell decomposition of the searchlight configuration space ($\mathbb{T}^N$).

Definition 8 (Critical Angle). An angle ψ is a *critical angle*[b] of $s^{[i]}$ if $\theta^{[i]} = \psi$

[b]Thanks to Howie Choset for pointing out the appropriateness of the name "critical angle". Taking the origin at a searchlight, its beam can be viewed as a level set of the Morse function $h(x, y) = \tan(y/x)$ so that the critical angles are where $h(x, y)$ has a critical point (constituted by a reflex vertex of $\partial\mathcal{E}$ or a searchlight in the interior of $\mathcal{E}$). See Ref. 19, 20.

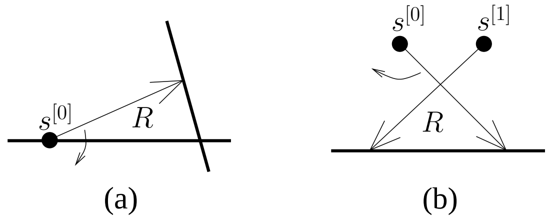

Fig. 2. How a maximal nonseparable region R can be made to disappear by a single searchlight $s^{[0]}$ rotating as indicated in each case by the smaller arrow. The thick line segments may represent either portions of $\partial\mathcal{E}$ or other searchlight beams. In this way, (a) could depict $s^{[0]}$ rotating either into $\partial\mathcal{E}$, or into coincidence with another searchlight's beam. Likewise, (b) could depict $s^{[0]}$ rotating either onto the intersection of other searchlights' beams, or onto the intersection point of another searchlight's beam with $\partial\mathcal{E}$.

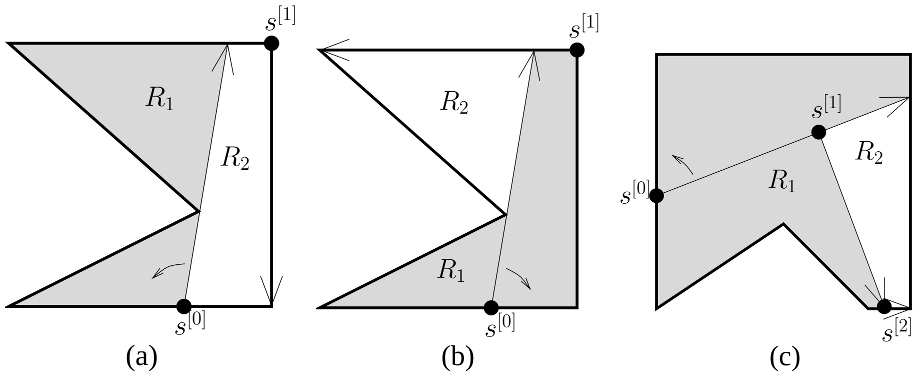

Fig. 3. The only manner in which two or more maximal nonseparable regions can merge into one is by some searchlight rotating either (a) past a reflex vertex of $\partial\mathcal{E}$ where its visibility is occluded, (b) away from a reflex vertex which it was grazing, or (c) past another searchlight not on $\partial\mathcal{E}$. In each example the clear region R_1 will become contaminated when it merges with the contaminated region R_2. Note that in (a) and (b) the reflex vertex could have also been a flat edge (between two reflex vertices) aligned with $s^{[0]}$'s beam, but the effect is the same.

implies either

(i) $s^{[i]}$ is located on an edge (including endpoints) of $\partial\mathcal{E}$ and is aimed along that edge,
(ii) $s^{[i]}$ is aimed at one of its visibility gaps,
(iii) $s^{[i]}$ is aimed at another visible searchlight, or
(iv) $s^{[i]}$ is aimed directly away from another visible searchlight.

Basic examples of critical angles are shown by the dashed lines in Fig. 4. More complicated examples can be found in Figs. 10 and 11. A searchlight configuration is a *critical configuration* if every searchlight is aimed along one of its critical angles. An information state corresponding to searchlights in a critical configuration is a

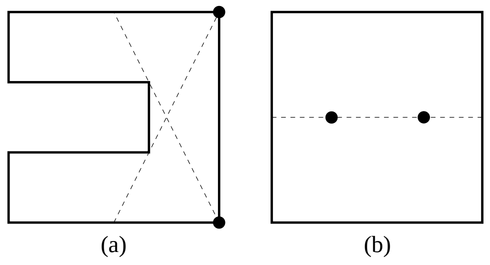

Fig. 4. Examples of critical angles: (a) Each searchlight has two critical angles aiming along the adjacent walls and one aiming towards its visibility gap. (b) Each searchlight has two critical angles, one pointing directly toward the other searchlight and one pointing directly away.

critical information state.

The next definition gives a useful notation for expressing the relationship between two information states with the same searchlight configuration but different contamination states.

Definition 9 (Partial Ordering on Information States). Given two information states (Θ_1, C_1) and (Θ_2, C_2), we write $(\Theta_1, C_1) \succeq (\Theta_2, C_2)$ if $\Theta_1 = \Theta_2$ and every maximal nonseparable region which is clear in C_1 is also clear in C_2. If it is understood from context that $\Theta_1 = \Theta_2$, then we simply write $C_1 \succeq C_2$.

Definition 10 (Critical Intervals and Rectangles). Let $\psi_j^{[i]}$ and $\psi_k^{[i]}$ be either adjacent or equal critical angles of $s^{[i]}$. Then an angular interval $\lambda^{[i]} =]\psi_j^{[i]}, \psi_k^{[i]}[\subset \mathbb{T}$ (resp. $[\psi_j^{[i]}, \psi_k^{[i]}]$) consisting of all angles which are

(i) between $\psi_j^{[i]}$ and $\psi_k^{[i]}$, and
(ii) ccw from $\psi_j^{[i]}$ to $\psi_k^{[i]}$

is an *open critical interval* (resp. *closed critical interval*) if $\lambda^{[i]}$ does not contain any critical angles of $s^{[i]}$ (resp. other than $\psi_j^{[i]}$ and $\psi_k^{[i]}$). The angles $\psi_j^{[i]}$ and $\psi_k^{[i]}$ are the *bounding angles* of the critical interval. In the case $\psi_j^{[i]} = \psi_k^{[i]}$, $\lambda^{[i]} = \{\psi_j^{[i]}\}$ and $\lambda^{[i]}$ is called a *null critical interval.* The Cartesian product $\Lambda = \lambda^{[0]} \times \cdots \times \lambda^{[N-1]} \subset \mathbb{T}^N$ of critical intervals is a *critical rectangle.*

Definition 11 (Minimal Critical Rectangle). Given a searchlight configuration $\Theta_0 = \{\theta_0^{[0]}, \ldots, \theta_0^{[N-1]}\}$, the *minimal critical rectangle* Λ containing Θ_0 is the Cartesian product of critical intervals $\Lambda = \lambda^{[0]} \times \cdots \times \lambda^{[N-1]}$, where each $\lambda^{[i]}$ $(i = 0, \ldots, N-1)$ is the unique smallest critical interval containing $\theta_0^{[i]}$. If $\theta_0^{[i]}$ is a noncritical angle, then $\lambda^{[i]}$ is open. If $\theta_0^{[i]}$ is critical, then $\lambda^{[i]}$ is a null (closed) critical interval.

Remark 1. By Definition 11, the minimal critical rectangle Λ containing a searchlight configuration Θ_0 is not necessarily closed or open. In particular, Λ is

(i) closed iff Θ_0 is a critical configuration,
(ii) open iff in Θ_0 no searchlight is aimed at a critical angle, or
(iii) neither open nor closed otherwise.

Lemma 1. *Given a searchlight configuration Θ_0, the minimal critical rectangle Λ containing Θ_0 is unique.*

Proof. Letting $\Lambda = \lambda^{[0]} \times \cdots \times \lambda^{[N-1]}$ as in Definition 11, uniqueness of Λ follows immediately from the uniqueness of each $\lambda^{[i]}$ $(i = 0, \ldots, N-1)$. □

If the searchlight configuration does not leave a minimal critical rectangle, then by definition no merging can happen[c]. This gives the following lemma.

Lemma 2. *Let Λ be the minimal critical rectangle containing some searchlight configuration. Then any maximal nonseparable region which is cleared without leaving Λ necessarily remains clear until Λ is left.*

In fact we can make a slightly stronger statement as in the following two definitions and lemma.

Definition 12 (Λ-equivalence of maximal nonseparable regions). Let Λ be the minimal critical rectangle containing some searchlight configuration, and R and R' distinct maximal nonseparable regions present when the searchlight configuration is somewhere in $\bar{\Lambda}$. Then R and R' are Λ-*equivalent* if it is possible for R and R' to merge without the searchlight configuration leaving $\bar{\Lambda}$.

Definition 13 (modulo Λ-equivalence). Let Λ be the minimal critical rectangle containing some searchlight configuration. When we say a property holds for maximal nonseparable regions *modulo Λ-equivalence*, we intend the following. Any Λ-equivalence class of maximal nonseparable regions, say $\{R_0, R_1, \ldots, R_{M-1}\}$, is to be interpreted as a single maximal nonseparable region, say R, where

(i) the contamination state of R is taken to be the logical OR of the contamination states of $R_0, R_1, \ldots, R_{M-1}$, and
(ii) the area of R is taken to be the sum of the areas of $R_0, R_1, \ldots, R_{M-1}$.

Definitions 12 and 13 are illustrated in Fig. 5. Roughly put, making a statement about maximal nonseparable regions modulo Λ-equivalence amounts to ignoring the splitting and merging which can happen when the searchlight configuration moves between the minimal critical rectangle Λ and its relative boundary $\bar{\Lambda} \setminus \Lambda$.

Lemma 3. *Let Λ be the minimal critical rectangle containing some searchlight configuration. Then, modulo Λ-equivalence, any maximal nonseparable region which is cleared without leaving $\bar{\Lambda}$ necessarily remains clear until $\bar{\Lambda}$ is left.*

[c]In this way, critical rectangles are reminiscent of the "conservative regions" defined in Ref. 15.

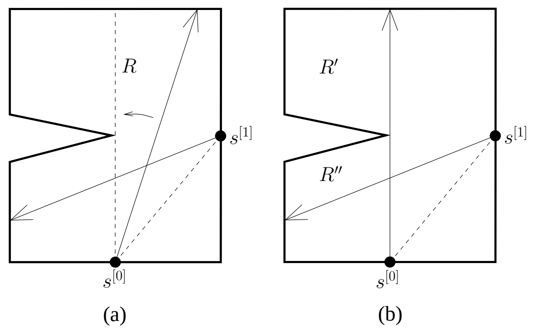

Fig. 5. This example illustrates Definitions 12 and 13. Searchlights $s^{[0]}$ and $s^{[1]}$ have critical angles along their walls and the dashed lines. Let Λ be the minimal critical rectangle containing the noncritical configuration shown in (a). If the configuration moves to the relative boundary $\bar{\Lambda} \setminus \Lambda$ of Λ as in (b), then the maximal nonseparable region R splits into R' and R''. Likewise, if the configuration were to move back to the relative interior of Λ, then R' and R'' could merge back into R. The maximal nonseparable regions R' and R'' are thus Λ-equivalent. To consider a maximal nonseparable region R modulo Λ-equivalence means to ignore the splitting and merging which occurs when the configuration enters $\bar{\Lambda} \setminus \Lambda$. In this way, R would be identified here with the union of R' and R'' and the contamination state of R would be 1 if and only if it were 1 for either R' or R''.

Proof. A maximal nonseparable region can only be recontaminated by merging with a contaminated maximal nonseparable region. However, if these two regions merged without the searchlight configuration leaving $\bar{\Lambda}$, then they must be Λ-equivalent and therefore should be interpreted as a single contaminated maximal nonseparable region. □

Now we are ready to define the classes of schedules we are interested in.

Definition 14 (Classes of Schedules). A schedule is called

(i) *sequential* if only one searchlight is active at a time,
(ii) *critical* if searchlights only rotate between critical angles, i.e., while rotating they never stop or change direction at a noncritical angle,
(iii) *rotation-monotone* if each searchlight is constrained to rotate either exclusively cw or exclusively ccw,
(iv) *contamination-monotone* if no point in the environment changes contamination state more than once, and
(v) *general* if it does not necessarily fall under any of the above special classes.

We have all but stated explicitly what the exact cell decomposition of the searchlights' joint configuration space ($\mathbb{T}^N$) is. The cells are precisely the set of all closed critical rectangles of positive measure, i.e., those which are the Cartesian product of non-null critical intervals (see example Fig. 6). This decomposition is exact in the sense that for the Searchlight Scheduling Problem it suffices to consider only the class of schedules which travel along cell boundaries, which is the class of critical

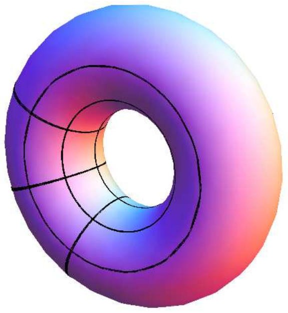

Fig. 6. For the simple problem instance of Fig. 4a, each of the 2 searchlights has 3 critical angles. Together, these 6 critical angles define the exact cell decomposition of the searchlight configuration space $\mathbb{T}^2$ (shown embedded in $\mathbb{R}^3$, thick black lines are the cell boundaries).

sequential schedules. This exactness will be captured rigorously in the main Theorems 1, 2, and Corollary 1 at the end of this section, but first we require a few more definitions and lemmas.

Definition 15 (Atomic and Critical Atomic Actions). Let $\mathcal{A}^{[i]}$ denote a rotation action by a searchlight $s^{[i]}$ from an angle α_{init} to an angle α_{fin}, where α_{init} and α_{fin} may or may not be critical angles. $\mathcal{A}^{[i]}$ is *atomic* if

(i) $s^{[i]}$ rotates monotonically cw or ccw without stopping, and
(ii) $s^{[i]}$ does not aim at any critical angle during the execution of the action except possibly α_{init} and/or α_{fin}.

If α_{init} and α_{fin} are both critical angles, then $\mathcal{A}^{[i]}$ is called a *critical atomic action.*

Simply put, a rotation action being atomic just amounts to the searchlight not changing direction of rotation nor crossing a critical angle.

Definition 16 (Projections of Atomic Actions). Given an atomic action $\mathcal{A}^{[i]}$ there is a minimal critical interval $\lambda^{[i]}$ containing that action. The *forward projection* of $\mathcal{A}^{[i]}$, denoted $\hat{\mathcal{A}}^{[i]}$, rotates $s^{[i]}$ over all of $\bar{\lambda}^{[i]}$ by rotating in the same direction as $\mathcal{A}^{[i]}$. The *reverse projection* of $\mathcal{A}^{[i]}$, denoted $\check{\mathcal{A}}^{[i]}$, rotates $s^{[i]}$ over all of $\bar{\lambda}^{[i]}$ by rotating in the direction opposing $\mathcal{A}^{[i]}$. See Fig. 7.

Lemma 4. *Any sequential searchlight schedule can be written as a discrete sequence of atomic rotation actions, i.e., a sequence of the form*

$$\{\mathcal{A}_m^{[i_m]}\}_{m=0}^{M-1} = \mathcal{A}_0^{[i_0]}\mathcal{A}_1^{[i_1]}\mathcal{A}_2^{[i_2]}\cdots\mathcal{A}_{M-1}^{[i_{M-1}]},$$

where each $\mathcal{A}_m^{[i_m]}$ denotes an atomic rotation action by searchlight i_m.

Proof. Given a sequential schedule represented by a sequence of actions, any non-atomic actions can be broken up into an appropriate number of atomic actions by

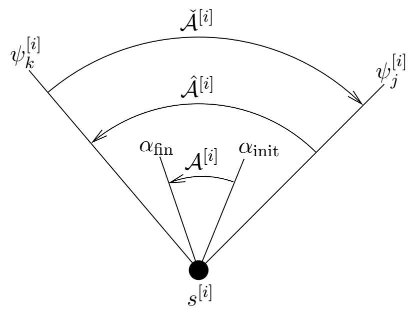

Fig. 7. The forward projection $\hat{\mathcal{A}}^{[i]}$ and reverse projection $\check{\mathcal{A}}^{[i]}$ of an atomic action $\mathcal{A}^{[i]}$. $\psi_j^{[i]}$ and $\psi_k^{[i]}$ signify the bounding critical angles of the minimal critical interval ($\lambda^{[i]}$ in Definition 16) containing $\mathcal{A}^{[i]}$.

splitting at the instances of time when a searchlight rotates over a critical angle or changes rotation direction. □

Lemma 5 (Schedule Decomposition into Time Intervals). *For any general search schedule $\Theta(t)$, $t \in [0, T]$, there exists a unique finite increasing sequence of times $t_{0,0}, t_{0,1}, t_{0,2}, \ldots, t_{1,0}, t_{1,1}, t_{1,2}, \ldots, t_{2,0}, t_{2,1}, t_{2,2}, \ldots, t_{M,0}$, where*

(i) $t_{0,0} = 0$ *and* $t_{M,0} = T$,

(ii) the times $\{t_{i,j} | \ i \in \{1, \ldots, M-1\}$ *and* $j = 0\}$ *correspond one-to-one to the times other than* 0 *and* T *when there is a change in the minimal critical rectangle containing the searchlight configuration,*[d] *and*

(iii) the times $\{t_{i,j} | \ i \in \{0, \ldots, M-1\}$ *and* $j > 0\}$ *correspond one-to-one to the times when one or more contaminated maximal nonseparable regions disappear but there is not concurrently a change in the minimal critical rectangle containing the searchlight configuration.*

Proof. Recall there are finitely many searchlights, each searchlight only has a finite number of critical angles, and searchlights may change direction of rotation only a finite number of times. From these observations, it is clear that the minimal critical rectangle containing the searchlight configuration can only change a finite number of times, i.e., M is finite. Finiteness of the subsequences $t_{i,1}, t_{i,2}, t_{i,3}, \ldots$ (for $i \in \{0, 1, \ldots, M-1\}$) follows from the facts that (i) there are only finitely many maximal nonseparable regions, and (ii) while the searchlight configuration remains in the same minimal critical rectangle, any contaminated maximal nonseparable regions which disappear cannot be recontaminated (see Lemma 2). Uniqueness follows from the one-to-one correspondence between times and events. □

[d] At $t_{0,0} = 0$ and $t_{M,0} = T$ there may or may not be a change in the minimal critical rectangle containing the searchlight configuration.

Lemma 6. *Let Λ be the minimal critical rectangle containing some searchlight configuration. Then, modulo Λ-equivalence, the area of each maximal nonseparable region is a continuous function of the searchlight configuration when restricted to $\bar{\Lambda}$.*

Proof. We know from elementary geometry that the area of a polygon is a continuous function of the coordinates of its vertices. Maximal nonseparable regions are polygons whose vertices have the following property: the coordinates of the vertices are continuous functions of the searchlight configuration. □

Lemma 7. *Suppose a maximal nonseparable region R can be made to disappear by a single atomic action $\mathcal{A}^{[i]}$ from a configuration Θ_0 (cf Fig. 2). Let Θ_1 be a configuration of the searchlights identical to Θ_0 except that $s^{[i]}$ is aimed at an angle in the interior of the angular interval over which it sweeps according to $\mathcal{A}^{[i]}$, and let Λ be the minimal critical rectangle containing Θ_1.*[e] *Then the area of R is a monotonic continuous function of each supporting searchlight's configuration (holding all other searchlights' configurations fixed) while the configuration is restricted to $\bar{\Lambda}$.*

Proof. Observe that R cannot extend around a hole of $\mathcal{E}$, otherwise some portion of R would be invisible to $s^{[i]}$. Also, there can be no searchlights located in the interior of R, otherwise $s^{[i]}$ would have to rotate over another searchlight to clear R (which would mean $\mathcal{A}^{[i]}$ were not atomic). From these properties it follows, that while the searchlights remain in $\bar{\Lambda}$, two facts hold true: (i) the portion of R's boundary formed by its supporting searchlight beams must be convex[f], and (ii) R must lie entirely on one side of each supporting searchlight's beam. If any supporting beam rotates towards the interior of R, then R's area must decrease. Likewise the area increases if any supporting beam rotates away from the interior, so we have established monotonicity. Continuity follows as in Lemma 6. □

Theorem 1 (Reduction from General to Sequential). *Any instance of the Searchlight Scheduling Problem which permits a general search schedule also permits a sequential search schedule.*

Proof. Let $\Theta(t)$, $t \in [0, T]$, be a general search schedule and

$$t_{0,0}, t_{0,1}, t_{0,2}, \ldots, t_{1,0}, t_{1,1}, t_{1,2}, \ldots, t_{2,0}, t_{2,1}, t_{2,2}, \ldots, t_{M,0}$$

a sequence of times as described in Lemma 5. We say that a schedule $\tilde{\Theta}(t)$ *emulates* another schedule $\Theta(t)$ over a time interval $[t_{\text{init}}, t_{\text{fin}}]$ if $\tilde{\Theta}(t_{\text{init}}) = \Theta(t_{\text{init}})$ and

[e]The important feature of Λ is that if Λ' is the minimal critical rectangle containing Θ_0, then $\bar{\Lambda}' \subset \bar{\Lambda}$ and the searchlight configuration will remain in $\bar{\Lambda}$ while $s^{[i]}$ executes $\mathcal{A}^{[i]}$.

[f]By "convex" we mean that when two searchlight beams form adjacent edges of R, then the interior angle between those edges is less than π. The portion of R's boundary formed by the environment boundary may or may not be convex.

$\tilde{C}(t_{\text{init}}) = C(t_{\text{init}})$ implies $(\tilde{\Theta}(t_{\text{fin}}), \tilde{C}(t_{\text{fin}})) \preceq (\Theta(t_{\text{fin}}), C(t_{\text{fin}}))$. To prove the theorem it suffices to show a sequential schedule $\tilde{\Theta}(t)$ can be constructed which emulates $\Theta(t)$ over the time interval $[0, T]$.

We first show that $\Theta(t)$ can be emulated by a sequential schedule over each time interval of the form $[t_{i,0}, t_{i,0}+\epsilon]$, $i \in \{0, 1, \ldots, M-1\}$, where $\epsilon \in \,]0, t_{i,1} - t_{i,0}[$. This is when merging can occur (review Fig. 3). Suppose we simply execute a sequence of atomic actions $\mathcal{A}_0^{[0]}\mathcal{A}_1^{[1]}\mathcal{A}_2^{[2]}\cdots\mathcal{A}_{N-1}^{[N-1]}$ which rotates each searchlight directly from $\theta^{[i]}(t_{i,0})$ to $\theta^{[i]}(t_{i,0}+\epsilon)$, $i \in \{1, \ldots, N\}$, one at a time in no particular order. It suffices to guarantee such a sequence of atomic actions does not cause any maximal nonseparable region, say R, to merge with a contaminated region, say R', which R did not merge with under $\Theta(t)$.[g] This is indeed the case, for suppose that R, through $\mathcal{A}_0^{[0]}\mathcal{A}_1^{[1]}\mathcal{A}_2^{[2]}\cdots\mathcal{A}_{N-1}^{[N-1]}$, does merge with such an R'. This is only possible if the support of R changes to include a new beam that will cause the merge. The only way to change the support of R in this manner is for one or more supporting beams of R to cross the intersection of a combination of other searchlights' beams and $\partial\mathcal{E}$. In such a case, it must be that R' just appeared as an artifact of using the sequence of atomic actions, thus R' is clear (see examples in Fig. 8).

We next consider the existence of a sequential schedule to emulate $\Theta(t)$ only during a time interval of the form $[t_{i,0}+\epsilon, t_{i+1,0}]$, $i \in \{0, 1, \ldots, M-1\}$, where $\epsilon \in \,]0, t_{i,1} - t_{i,0}[$. Let Λ be the minimal critical rectangle containing $\Theta(t_{i,0}+\epsilon)$. Recall that a maximal nonseparable region disappears if and only if its area goes to zero and this area, modulo Λ-equivalence, is a continuous function of the searchlight configuration when restricted to $\bar{\Lambda}$ (see Lemma 6). Together with Lemmas 3, this implies that all contaminated maximal nonseparable regions which are caused to disappear by $\Theta(t)$ during $(t_{i,0}, t_{i+1,0})$ can also be made to disappear one at a time by a sequential schedule. In particular, $\Theta(t)$ can be emulated over $[t_{i,0}+\epsilon, t_{i+1,0}]$ by any sequential schedule $\tilde{\Theta}(t)$ which visits successively the configurations $\Theta(t_{i,0}+\epsilon), \Theta(t_{i,1}), \Theta(t_{i,2}), \ldots, \Theta(t_{i+1,0})$ without leaving $\bar{\Lambda}$.[h]

We now know that the general schedule $\Theta(t)$ can be emulated by a sequential schedule $\tilde{\Theta}(t)$ over each interval $[t_{i,0}, t_{i+1,0}]$, $i \in \{0, 1, \ldots, M-1\}$, such that $\tilde{\Theta}(t_{i+1,0}) = \Theta(t_{i+1,0})$. Therefore concatenating the sequential schedules produces a sequential search schedule $\tilde{\Theta}(t)$ emulating $\Theta(t)$ over the duration $[0, T]$. □

Theorem 2 (Reduction from Sequential to Critical Sequential). *Any instance of the Searchlight Scheduling Problem which permits a sequential search schedule also permits a critical sequential search schedule.*

Proof. By Lemma 4 it suffices to show that given an atomic sequential schedule, i.e., a search schedule written as a sequence of atomic actions $\{\mathcal{A}_m^{[i_m]}\}_{m=0}^{M-1} =$

[g] If multiple regions merge into one under $\Theta(t)$, then these regions may instead merge one at a time when using $\mathcal{A}_0^{[0]}\mathcal{A}_1^{[1]}\mathcal{A}_2^{[2]}\cdots\mathcal{A}_{N-1}^{[N-1]}$, but this does not affect our line of argument.

[h] In our definition of emulation, it is only important to visit $\Theta(t_{i,0}+\epsilon)$ first and $\Theta(t_{i+1,0})$ last, otherwise the order of visiting the configurations does not matter.

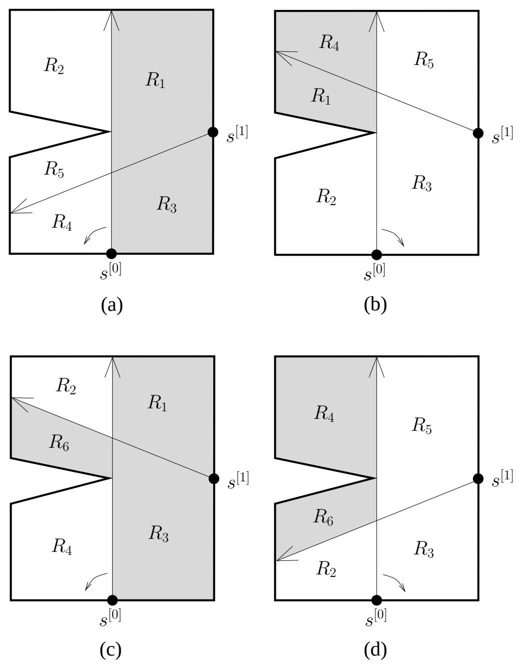

Fig. 8. Gray regions are clear. The effect of a searchlight $s^{[0]}$ executing the action indicated by the smaller arrow is (a) R_1 merges with R_2, (b) R_1 merges with R_2, (c) R_1 merges with R_2 and R_3 merges with R_6, (d) R_4 merges with R_6. (a) and (b) are analogous to (a) and (b), resp., of Fig. 3. On the other hand, (c) and (d) show the effects of $s^{[1]}$ rotating over the intersection of $s^{[0]}$'s beam with the reflex vertex, before $s^{[0]}$'s action is executed as in (a) and (b), respectively. Note that R_6 in (c) and (d) are clear because they are simply artifacts which appeared as a result of $s^{[1]}$ rotating over the intersection of $s^{[0]}$'s beam with the reflex vertex.

$\mathcal{A}_0^{[i_0]}\mathcal{A}_1^{[i_1]}\mathcal{A}_2^{[i_2]}\cdots\mathcal{A}_{M-1}^{[i_{M-1}]}$, we can construct another search schedule expressed as a sequence of critical atomic actions. Reducing the problem further, suppose that from an atomic sequential search schedule $\{\mathcal{A}_m^{[i_m]}\}_{m=0}^{M-1}$ we are able to construct a new atomic sequential search schedule $\{\tilde{\mathcal{A}}_m^{[i_m]}\}_{m=0}^{\tilde{M}-1}$ such that for an arbitrary searchlight, say $s^{[0]}$,

(i) the actions executed by searchlights other than $s^{[0]}$ are unaltered, and
(ii) the actions executed by $s^{[0]}$ are exclusively critical atomic (though they may increase in number).

If we can find a procedure to construct such a schedule, then this procedure could be repeated for each $s^{[i]}$, $i \in \{0, \ldots, N-1\}$, until we are left with a critical atomic sequential schedule. We show such a procedure exists.

Let (Θ_m, C_m) denote the information state of the original schedule $\{\mathcal{A}_m^{[i_m]}\}_{m=0}^{M-1}$ just before the mth action is executed. Without loss of generality, assume that in $\{\mathcal{A}_m^{[i_m]}\}_{m=0}^{M-1}$ all searchlights are initially aimed at critical angles. We construct from $\{\mathcal{A}_m^{[i_m]}\}_{m=0}^{M-1}$ another search schedule $\{\tilde{\mathcal{A}}_m^{[i_m]}\}_{m=0}^{\tilde{M}-1}$ as described above essentially by modifying $\{\mathcal{A}_m^{[i_m]}\}_{m=0}^{M-1}$ action by action. Suppose $\mathcal{A}_k^{[0]}$ is the first action which rotates $s^{[0]}$ from a critical angle $\psi_1^{[0]}$ to a noncritical angle α contained in the critical interval $[\psi_1^{[0]}, \psi_2^{[0]}]$. We can let $\tilde{\mathcal{A}}_m^{[i_m]} = \mathcal{A}_m^{[i_m]}$, $m \in \{0, \ldots, k-1\}$, but we do not want $s^{[0]}$ to stop at a noncritical angle, so we set $\tilde{\mathcal{A}}_k^{[0]} = \hat{\mathcal{A}}_k^{[0]}$ (recall Definition 16). Since $\tilde{\mathcal{A}}_k^{[0]}$ sweeps over a larger region than $\mathcal{A}_k^{[0]}$ and does not cause any merging which $\mathcal{A}_k^{[0]}$ does not, it must clear the same maximal nonseparable regions as $\mathcal{A}_k^{[0]}$. We only need to worry about how this change in $s^{[0]}$'s configuration will effect subsequent actions by other searchlights. Suppose the next action is $\mathcal{A}_{k+1}^{[1]}$. We keep $\tilde{\mathcal{A}}_{k+1}^{[1]} = \mathcal{A}_{k+1}^{[1]}$, but since during $\tilde{\mathcal{A}}_{k+1}^{[1]}$ $s^{[0]}$ was aimed at $\psi_2^{[0]}$ (instead of α), the effect of $\tilde{\mathcal{A}}_{k+1}^{[1]}$ may be different than $\mathcal{A}_{k+1}^{[1]}$. In particular, $\tilde{\mathcal{A}}_{k+1}^{[1]}$ may not clear all the same maximal nonseparable regions which $\mathcal{A}_{k+1}^{[1]}$ did. The monotonicity property of Lemma 7 guarantees we can compensate for this difference simply by choosing $\tilde{\mathcal{A}}_{k+2}^{[0]} = \check{\mathcal{A}}_k^{[0]}$. The important result of executing such an $\tilde{\mathcal{A}}_{k+2}^{[0]}$ is that if we were to then rotate $s^{[0]}$ from $\psi_1^{[0]}$ directly back to α (though we do not actually do this because we only want $s^{[0]}$ to stop at critical angles), then we would end up in the configuration Θ_{k+2} with contamination state $C \preceq C_{k+2}$. So far we have constructed a schedule $\{\tilde{\mathcal{A}}_m^{[i_m]}\}_{m=0}^{k+2}$ from $\{\mathcal{A}_m^{[i_m]}\}_{m=0}^{k+1}$. The procedure continues alternately taking an action from $\{\mathcal{A}_m^{[i_m]}\}_{m=k+2}^{M}$ and rotating $s^{[0]}$ over the critical interval $[\psi_1^{[0]}, \psi_2^{[0]}]$. After every pair of such actions, the monotonicity property of Lemma 7 guarantees the same maximal nonseparable regions will be cleared as in the original schedule. In the end we are left with a sequential search schedule $\{\tilde{\mathcal{A}}_m^{[i_m]}\}_{m=0}^{\tilde{M}-1}$ which satisfies the above enumerated requirements. □

Putting together Theorems 1 and 2, we finally arrive at the main solution space reduction result.

Corollary 1 (Main Reduction Result). *Any instance of the Searchlight Scheduling Problem which permits a general search schedule also permits a critical sequential search schedule.*

Proof. Immediate from combining Theorems 1 and 2. □

An interesting and open problem is whether it is possible to reduce the solution space even further, e.g., as in Conjecture 3.1 below. Further reduction of the solution space could allow for faster computation of search schedules.

Conjecture 3.1 (Rotation- and Contamination-Monotonicity). *Any instance of the Searchlight Scheduling Problem which permits a general search schedule also permits a rotation- and contamination-monotone critical sequential search schedule.*[i]

[i]It is intended that there would exist a rotation- and contamination-monotone critical sequential

Table 1. Geometric Preprocessing

```
Input:  geometric problem instance $(\mathcal{E}, P)$
1: for all searchlights $i = 0, \ldots, N-1$ do
2:     compute visibility polygon $\mathcal{V}(p^{[i]})$;
3:     extract critical angles $\Psi^{[i]} = \{\psi_0^{[i]}, \psi_1^{[i]}, \ldots, \psi_{n_\psi^{[i]}-1}^{[i]}\}$ from $\mathcal{V}(p^{[i]})$;
4: compute cells $\gamma_0, \gamma_1, \ldots, \gamma_{n_\gamma - 1}$ of environment partition $\Gamma$;
5: compute partition dual graph $\mathcal{G}_\Gamma$;
```

4. A Complete Algorithm

The solution space reduction result Corollary 1 tells us that if we can systematically search the space of critical sequential schedules, then we are guaranteed to find a search schedule if one exists. Our algorithm does just this by searching for a solution trajectory through a discretization of the information space. Every critical sequential search schedule, by definition, can be represented by a sequence of critical atomic actions connecting critical information states (as in, e.g., Fig. 12), thus the information space discretization is defined as follows.

Definition 17 (Directed Information Graph $\mathcal{G}_\mathcal{I}$).

(i) nodes correspond to critical information states, and
(ii) there is a directed edge from one node, say x, to another node, say x', if and only if it is possible to reach x' from x by executing a single critical atomic action.

If a search schedule exists, then our algorithm will find it by searching $\mathcal{G}_\mathcal{I}$ for a path from a completely contaminated node to a completely clear node. In its entirety the algorithm consists of two parts: (i) geometric preprocessing which extracts combinatorial information from the problem instance geometry, followed by (ii) systematic search of the information graph. We detail these parts separately in Sections 4.1 and 4.2, then provide a discussion of implementation and computed examples in Section 4.3.

4.1. *Geometric Preprocessing*

The geometric preprocessing, taking the environment geometry and searchlight positions as input, computes an environment partition and a graph by means of the sequence of computations shown in Table 1. Each of the searchlights' visibility polygons $\mathcal{V}(p^{[i]})$, $i \in \{0, \ldots, N-1\}$, will have at most n edges and can be computed in $\mathcal{O}(n \log n)$ time.[21] Let $\Psi = \{\Psi^{[0]}, \Psi^{[1]}, \ldots, \Psi^{[N-1]}\}$, where $\Psi^{[i]} = \{\psi_0^{[i]}, \psi_1^{[i]}, \ldots, \psi_{n_\psi^{[i]}-1}^{[i]}\}$ denotes a list of the ith searchlight's critical angles.

search schedule from some initial condition, not necessarily from any initial condition.

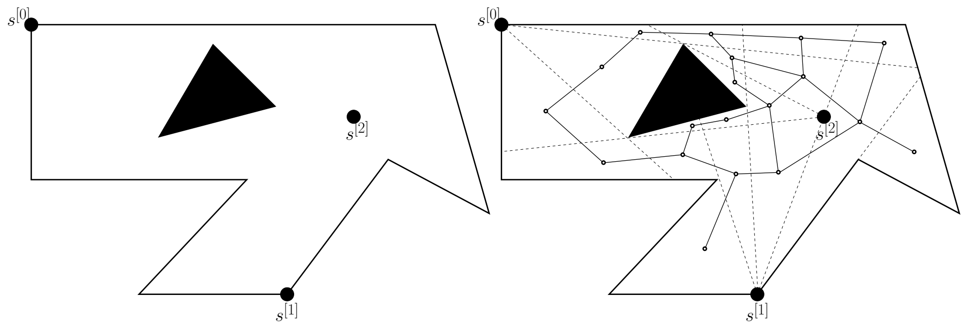

Fig. 9. The geometric preprocessing part of the complete algorithm (Tab. 1, Section 4) is illustrated using a simple problem instance having three searchlights and one hole. First a geometric description of the problem instance is taken as input (left). This consists of the environment geometry $\mathcal{E}$ together with the searchlight locations P. Next (right), the critical angles Ψ of the searchlights (dashed lines) and environment partition Γ are computed. Each cell of Γ is either completely clear or completely contaminated on any node of the information graph $\mathcal{G}_{\mathcal{I}}$. Finally, the dual graph $\mathcal{G}_{\Gamma}$ of the environment partition is computed. This instance was solved by our C++ implementation of the complete algorithm. The computed solution is illustrated in Fig. 12, statistics in Table 3.

Using Definition 8, each $\Psi^{[i]}$ can easily be extracted from $\mathcal{V}(p^{[i]})$ by checking for (i) radially aligned edges of $\mathcal{V}(p^{[i]})$ in $\mathcal{O}(n)$ time, and (ii) inclusion of other searchlights in $\mathcal{V}(p^{[i]})$ in $\mathcal{O}(Nn)$ time (point-in-polygon test).[22] Adding these time complexities gives a bound on the total time to compute Ψ.

Lemma 8. *The critical angles Ψ can be computed in $\mathcal{O}(Nn\log n + N^2n)$ time.*

Lemma 9 gives an upper bound on the size of each $\Psi^{[i]}$.

Lemma 9. *A searchlight can have no more than $r + 2N$ critical angles if located on $\partial\mathcal{E}$, otherwise no more than $r + 2N - 2$ if located in $\mathcal{E}^{\circ}$.*

Proof. These bounds follow directly from the Definition 8 of critical angles. A searchlight located on $\partial\mathcal{E}$ may have 2 critical angles due to adjacent edges of $\partial\mathcal{E}$, $2(N-1)$ due to line-of-sight with other searchlights, and r due to reflex vertices of $\mathcal{E}$, hence $2 + 2(N-1) + r = r + 2N$. For a searchlight located in $\mathcal{E}^{\circ}$ the only difference is that there can be no critical angles due to adjacent edges of $\partial\mathcal{E}$. □

In most instances, however, the number of critical angles is far less than the bound in Lemma 9.

We now describe how the environment partition Γ is constructed. For each searchlight $s^{[i]}$, $i \in \{0, \dots, N-1\}$, and critical angle $\psi_j^{[i]}$, $j \in \{0, \dots, n_{\psi}^{[i]} - 1\}$, the jth *critical segment* of the ith searchlight is the closed line segment consisting of all points illuminated by $s^{[i]}$ when aimed in the direction $\psi_j^{[i]}$. The critical segments together partition $\mathcal{E}$ into a finite set of simply connected polygonal *cells*.[j]

[j]Recall that in Section 3 we spoke of cells as part of an exact cell decomposition of the searchlight

Examples are shown by the dashed lines in Figs. 9, 10, 11. Representing $\mathcal{E}$ as a list of cells $\Gamma = \{\gamma_0, \gamma_1, \dots, \gamma_{n_\gamma - 1}\}$ allows one to encode the contamination state by a binary n_γ-tuple C_Γ. In C_Γ, each cell is labeled either "0" for clear or "1" for contaminated. This representation of a contamination state, together with the searchlights' joint configuration, is used to represent a node of $\mathcal{G}_\mathcal{I}$. Lemma 10 gives an upper bound on the time to compute Γ.

Lemma 10. *The cells $\gamma_0, \gamma_1, \dots, \gamma_{n_\gamma - 1}$ of the environment partition Γ can be computed in $\mathcal{O}((n + Nr + N^2)^8)$ time.*

Proof. Constructing the cells of Γ amounts to computing the faces of an arrangement of line segments. The faces of an arrangement of L line segments can be computed in $\mathcal{O}(L^8)$ time. We count the line segments contributing to our arrangement. There are n segments due to $\mathcal{E}$, and from Lemma 9 at most $N(r+2N)$ critical segments. Substituting $L = n + Nr + 2N^2$, we find the cells can be computed in $\mathcal{O}((n + Nr + 2N^2)^8)$ time. Because arrangements constitute an already well studied area of computational geometry, and for the sake of brevity, we omit further detail.[23] □

Lemma 11 gives an upper bound on the number of cells in Γ.

Lemma 11. *The number n_Γ of cells in the environment partition Γ is $\mathcal{O}(N^4 + r^2N^2)$.*

Proof. Given and arrangement of $L \in \mathbb{N}$ lines in the plane, the maximum number of regions they partition the plane into is $\frac{L(L+1)}{2} + 1$; see Ref. 23. According to Lemma 9, the interior of $\mathcal{E}$ is partitioned by at most $N(r+2N) = rN + 2N^2$ critical segments. Setting $L = rN + 2N^2$, we see that the affine hulls of the critical segments together partition the plane into at most

$$\frac{(rN + 2N^2)(rN + 2N^2 + 1)}{2} + 1 = \frac{r^2N^2 + 4rN^3 + rN + 4N^4 + 2N^2}{2} + 1$$

regions. This gives a conservative upper bound on the order of n_Γ. We did not have to take into account the n line segments of $\mathcal{E}$ because only the r reflex vertices can cause the cell count to increase (by one each), so the order would not have been affected. □

The final task of geometric preprocessing is to compute the undirected dual graph of Γ.

configuration space. These cells on $\mathbb{T}^N$ were a theoretical tool for obtaining the main reduction result Corollary 1. Since as part of the geometric preprocessing we partition the environment $\mathcal{E}$ into polygonal cells, we stipulate now that, whenever the term "cells" appears in Section 4, it is always the environment partition cells which are intended, not the cells of the exact cell decomposition of $\mathbb{T}^N$.

Definition 18 (Dual Partition Graph). The dual graph $\mathcal{G}_\Gamma$ of Γ is the undirected graph defined as follows:

(i) its nodes are the polygonal cells $\{\gamma_0, \gamma_1, \ldots, \gamma_{n_\gamma - 1}\}$ of Γ, and
(ii) there is a (undirected) edge from one node, say γ, to another node, say γ', if and only if the polygons γ and γ' share an edge.

An example of a partition dual graph is shown in Fig. 9. Encoding the adjacency information of the environment partition cells, $\mathcal{G}_\Gamma$ will later allow us to compute the recontamination which can occur during an information state transition. In this way, $\mathcal{G}_\Gamma$ is a parameter of the information state transition function $f_{\mathcal{G}_\Gamma}(x, u)$ as we will see in Section 4.2 and Table 2. Lemma 12 gives an upper bound on the time to compute $\mathcal{G}_\Gamma$.

Lemma 12. *The dual graph $\mathcal{G}_\Gamma$ of the evironment partition Γ can be computed in $\mathcal{O}((N^4 + r^2N^2)^2(n + Nr + N^2)^2)$ time.*

Proof. $\mathcal{G}_\Gamma$ can be easily computed by comparing every edge of every cell. When an edge of two cells matches, then an edge is added between the respective nodes of $\mathcal{G}_\Gamma$. Lemma 11 tells us we must check all pairs of $\mathcal{O}(N^4 + r^2N^2)$ cells. Each cell has at most $n + Nr + 2N^2$ edges, so for each of the $\mathcal{O}((N^4 + r^2N^2)^2)$ pairs of cells, $\mathcal{O}((n+Nr+N^2)^2)$ edges must be compared. The total time complexity is therefore $\mathcal{O}((N^4 + r^2N^2)^2(n + Nr + N^2)^2)$. □

Together Lemmas 8, 10, and 12 tell us that the total time complexity of geometric preprocessing is polynomial in the problem instance parameters N, n, and r. It may be possible to compute the environment partition Γ and its dual graph $\mathcal{G}_\Gamma$ faster than our bounds suggest. However, we have not spent much effort optimizing the geometric preprocessing because, as indicated by computed examples (see, e.g., Table 3), the overall time complexity of our algorithm is dominated by the information graph search described in Section 4.2.

4.2. *Searching the Information Graph $\mathcal{G}_\mathcal{I}$*

The information graph $\mathcal{G}_\mathcal{I}$ can be searched using any systematic graph search algorithm such as breadth-first, Dijkstra, or A^*. Graph search algorithms are surveyed nicely in, e.g., Ref. 16 and 24. We use breadth-first for simplicity. A pseudocode is provided in Table 2, where our notation closely follows that used on page 33 of Ref. 16. Since $\mathcal{G}_\mathcal{I}$ is typically very large, containing many irrelevant and unreachable nodes, we do not precompute $\mathcal{G}_\mathcal{I}$, but instead nodes are added to the representation only as visited by the graph search. The search begins by pushing an initial information state x_0 onto the FIFO (First-In First-Out) queue Q. In x_0, the contamination state $C_\Gamma = (1, 1, 1 \ldots, 1)$ (environment completely contaminated) and the searchlight configuration may be chosen arbitrarily. At each iteration of the main loop a node x is popped off the queue Q, added to the search tree T, and its out-neighbors

Table 2. Breadth First Search of Information Graph $\mathcal{G}_\mathcal{I}$

x_{initial}	:=	initial $\mathcal{G}_\mathcal{I}$ node with $C_\Gamma = (1,1,1,\ldots,1)$
X_{clear}	:=	set of $\mathcal{G}_\mathcal{I}$ nodes with $C_\Gamma = (0,0,0,\ldots,0)$
Q	:=	FIFO queue of alive $\mathcal{G}_\mathcal{I}$ nodes
T	:=	search tree of dead $\mathcal{G}_\mathcal{I}$ nodes
U	:=	set of critical atomic actions
$f_{\mathcal{G}_\Gamma}(x,u)$	:=	information state transition function

```
1: $Q$.Insert($x_{\text{initial}}$);
2: while $Q$ not empty do
3:   $x \leftarrow Q$.PopFirst();
4:   $T$.Insert($x$);
5:   if $x \in X_{\text{clear}}$ then
6:     return SUCCESS;
7:   for all $u \in U$ do
8:     $x' \leftarrow f_{\mathcal{G}_\Gamma}(x,u)$;
9:     if $x' \notin Q$ and $x' \notin T$ then
10:      $Q$.Insert($x'$);
11: return FAILURE;
```

(in $\mathcal{G}_\mathcal{I}$) are computed. If an out-neighbor is a goal node, i.e., an information state having a completely clear environment ($C_\Gamma = (0,0,0,\ldots,0)$), then the algorithm returns SUCCESS. If an out-neighbor is not a goal node and it is not redundant (not already in Q or T), then it is added onto the queue. FAILURE is returned if Q becomes empty and no goal node has been found, which means no solution exists. When the algorithm does return SUCCESS, a critical sequential search schedule can be recovered by backtracing pointers through the search tree and storing the sequence of critical atomic actions.

Theorem 3 and Corollary 2 provide upper bounds on the size of $\mathcal{G}_\mathcal{I}$.

Theorem 3 (Number of Nodes in $\mathcal{G}_\mathcal{I}$). *The number of nodes in the information graph $\mathcal{G}_\mathcal{I}$ is*[k] $(2N+r)^N 2^{\mathcal{O}(N^4+N^2r^2)}$.

Proof. Follows from Lemmas 11 and 9. To count the number of unique contamination labelings we raise 2 to the bound on the number of cells. $(2N+r)^N$ is an upper bound on the number of unique critical searchlight configurations. □

Stirling's approximation says that $N^N \approx N!$, so the upper bound in Theorem 3 grows with N worse than $N!$. The following Corollary 2 of Theorem 3 shows, however, that if there is an upper bound k on the maximum number of critical angles any single searchlight has, then the number of nodes in $\mathcal{G}_\mathcal{I}$ can be bounded by a

[k] A function $g(x)$ is in the set $2^{\mathcal{O}(h(x))}$ if $\exists\ x_0$ and $c \in \mathbb{R}_{>0}$ such that $x > x_0 \implies g(x) < 2^{ch(x)}$. Note that $\mathcal{O}(2^{h(x)})$ is a proper subset of $2^{\mathcal{O}(h(x))}$.

function which is only exponential in N.

Corollary 2 (Number of Nodes in $\mathcal{G}_{\mathcal{I}}$). *Suppose that each searchlight in an instance has at most k critical angles. Then the number of nodes in the information graph $\mathcal{G}_{\mathcal{I}}$ is $k^N 2^{\mathcal{O}(N^2k^2+r)}$.*

Proof. The number of line segments forming the environment partition can be no greater than $Nk + n$, so setting $\xi = Nk + n$ in the formula in the proof of Lemma 11 and adding r gives an upper bound on the number of cells. To count the number of unique (binary) contamination labelings, we raise 2 to the bound on the number of cells. k^N is an upper bound on the number of unique critical searchlight configurations. The resulting upper bound on the number of nodes in $\mathcal{G}_{\mathcal{I}}$ is $k^N 2^{\frac{1}{2}N^2k^2+\frac{1}{2}Nk+r+1}$. □

The bounds on the size of $\mathcal{G}_{\mathcal{I}}$ given in Theorem 3 Corollary 2 could be used to derive a bound on the time complexity of the graph search, however we do not do this because we believe such bounds would be too loose. Instead, in Theorem 4 we derive a bound which reflects the output sensitivity of the computation time.

Theorem 4 (Time Complexity of $\mathcal{G}_{\mathcal{I}}$ Breadth-First Search). *Suppose that, for a particular problem instance $(\mathcal{E}, P)$ and initial information state x_0, there exists a search schedule consisting of M^* critical atomic actions, and M^* is the smallest such number. Then performing the breadth-first search of $\mathcal{G}_{\mathcal{I}}$ shown in Table 2 starting from x_0 takes time $\mathcal{O}((N^4 + r^2N^2)(2N)^{2M^*}))$.*

Proof. Referring to Table 2, observe that the number of possible critical atomic actions $|U| = 2N$ (each searchlight can rotate either cw or ccw). This means each node of $\mathcal{G}_{\mathcal{I}}$ has at most $2N$ out-neighbors. Taking $2N$ as the branching factor of the breadth-first search tree, and knowing the search will terminate at depth M^*, we see that $\mathcal{O}((2N)^{M^*})$ nodes will have been visited. To generate each node (except for x_0) an information state transition must be computed, which can be done in $\mathcal{O}(n_\Gamma)$ time using a technique called *floodfill*.[1] So far the total time complexity we have accounted for is $\mathcal{O}(n_\Gamma(2N)^{M^*})$. Additional complexity is added by each node being compared with every other node to avoid redundancy in the search tree. To check whether two information states are equal costs $\mathcal{O}(N + n_\Gamma)$ time because the searchlight configurations and contamination state of each cell must be compared. There are $\mathcal{O}((2N)^{2M^*})$ pairs of nodes, so the redundancy checks result in total time complexity $\mathcal{O}((N + n_\Gamma)(2N)^{2M^*})$. Using Lemma 11 to substitute $\mathcal{O}(n_\Gamma) = \mathcal{O}(N^4 + r^2N^2)$, we obtain total time complexity $\mathcal{O}((N^4 + r^2N^2)(2N)^{2M^*})$. □

[1]*Floodfill* is technique commonly used in computer graphics for painting connected regions of rasterized images.

4.3. *Implementation and Computed Examples*

We have implemented our algorithm, both geometric preprocessing and information graph breadth-first search, in C++ on a 2.33GHz i686 processor using the Standard Library and the VisiLibity library for visibility computations.[25] The VisiLibity Library, which we used only for the geometric preprocessing step, uses so-called ϵ-geometry for robustness (Ref. 26, 27). Table 3 shows statistics from the computed examples of Figs. 1, 9, 10, and 11. Fig. 12 shows a graphical illustration of the critical sequential search schedule computed for the instance in Fig. 9.

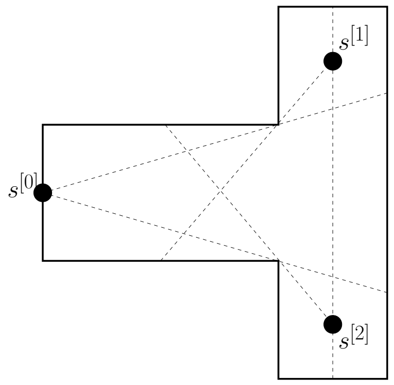

Fig. 10. Dashed lines show the critical angles for each searchlight and also partition the environment into discrete simply connected polygonal cells. Each cell is either completely clear or completely contaminated on any node of the information graph. This instance was solved by our C++ implementation of the complete algorithm described in Section 4. Computation statistics are found in Table 3.

We have tested dozens of problem instances and although the algorithm works well for instances with up to 4 guards and 5 critical angles per guard (such as the examples shown in Figs. 1, 9, 10, 11), it seems there is a very rapid combinatorial explosion for problem instances even slightly more complex. For example, one problem instance we tested had $n = 21$ vertices, $r = 14$ reflex vertices, $h = 3$ holes, $N = 5$ searchlights, and it computed for over 13 hours without finding a solution. This raises an important open question which we do not answer in this article, namely whether the general Searchlight Scheduling Problem is NP-hard.

5. Extension to Searchlights with Finite Field of View

A searchlight senses only along a ray, but many real sensors, such as security cameras, have a finite field of view. This motivates the definition of *ϕ-searchlight*,[m]

[m]The name "ϕ-searchlight" was inspired by the "ϕ-searchers" of Ref. 4, the difference being that a "ϕ-searcher" can rotate and translate, whereas a "ϕ-searchlight" can only rotate.

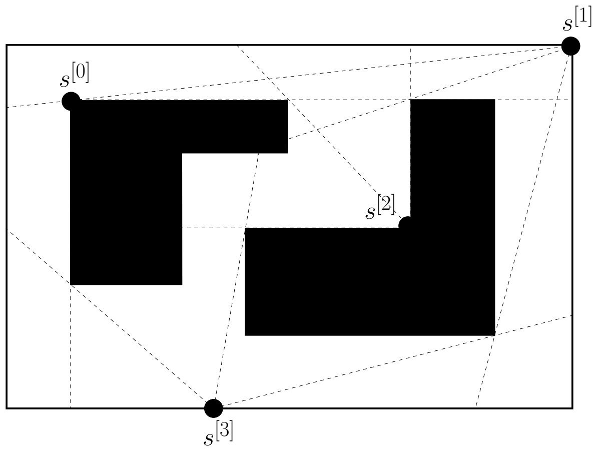

Fig. 11. Dashed lines show the critical angles for each searchlight and also partition the environment into discrete simply connected polygonal cells. Each cell is either completely clear or completely contaminated on any node of the information graph. This instance was solved by our C++ implementation of the complete algorithm described in Section 4. Computation statistics are found in Table 3.

Table 3. Statistics from computed examples.

Problem Instance	Guards	Edges in Environment	Cells in Environment Partition	Information Graph Nodes Visited	Geometric Preprocessing Time (seconds)	Information Graph Search Time (seconds)	Total Computation Time (seconds)
Fig. 1	2	8	6	13	< 0.01	< 0.01	< 0.01
Fig. 9, 12	3	11	19	497	0.03	0.02	0.05
Fig. 10	3	8	15	464	0.02	0.01	0.03
Fig. 11	4	16	22	5401	0.05	1.79	1.84

which is identical to a searchlight except instead of sensing only along a ray, it can sense anywhere within a finite field of view measured by an angle ϕ as illustrated in Fig. 13. Now we can define the *ϕ-Searchlight Scheduling Problem*:

Given N ϕ-searchlights with finite fields of view $\phi^{[0]}, \phi^{[1]}, \phi^{[2]}, \ldots, \phi^{[N]}$, find a rotation schedule such that any target in an environment will necessarily

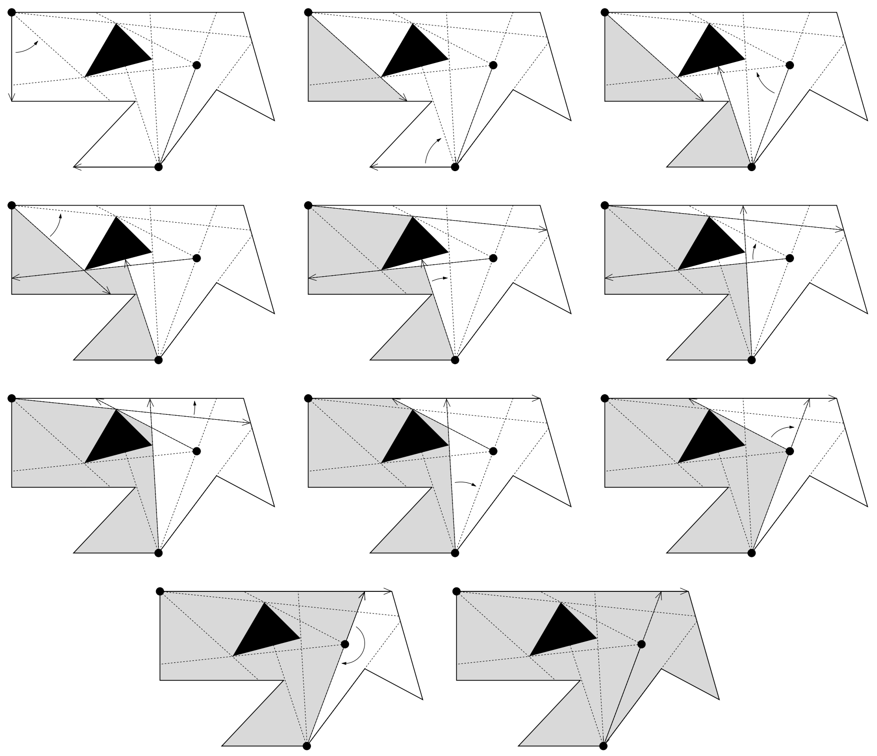

Fig. 12. From left to right, top to bottom, here is shown a critical sequential search schedule computed by our C++ implementation of the complete algorithm described in Section 4 (same instance as Fig. 9). Grey regions are clear, the smaller arrows indicate which searchlight will execute a critical atomic action at each step of the sequence. Computation statistics are found in Table 3.

be detected in finite time.

One must ask whether there is actually any advantage to having searchlights with finite field of view, e.g., whether there are problem instances which can be solved with ϕ-searchlights but not by searchlights. Indeed there are instances where greater fields of view enable a solution.

Lemma 13 (Increased Field of View Advantage). *As the field of view of ϕ-searchlights increases, the set of solvable problem instances grows.*

Proof. In the simple "hour-glass"[n] example of Fig. 14, there is no solution unless both searchlights have a field of view $\phi_{\min}$ or greater. $\square$

[n]Thanks to Nicola Ceccarelli for suggesting this example.

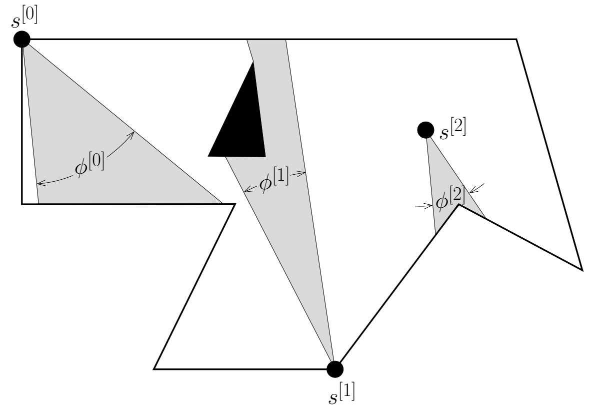

Fig. 13. In the ϕ-Searchlight Scheduling Problem, each searchlight may have a different field of view $\phi^{[0]}, \phi^{[1]}, \phi^{[2]}, \ldots, \phi^{[N]}$.

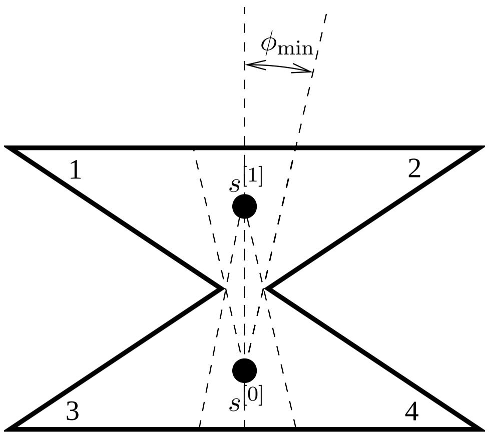

Fig. 14. In this simple "hour-glass" example, the environment cannot be cleared unless both ϕ-searchlights have field of view at least $\phi_{\min}$. To see why, imagine the fields of view are both less than $\phi_{\min}$ and notice if you clear any one of the corners (labeled 1, 2, 3, and 4), then trying to clear a second corner cannot be accomplished without contaminating the first. On the other hand, if both fields of view are $\phi_{\min}$ or greater, the problem is easily solvable because each ϕ-searchlight is able to simultaneously illuminate one of the corners and the other $\phi-$searchlight

The algorithm we have described in Section 4 could be used with ϕ-searchlights as is, but this would not take advantage of the added capabilities offered by greater fields of view. In fact only a small modification is necessary to obtain a complete algorithm for ϕ-searchlight scheduling. We need only redefine the critical angles. Observe that for a ϕ-searchlight, critical visibility events can only occur when there is a change in the set of searchlight critical angles (as per Definition 8) illuminated by its field of view. Therefore, taking the configuration $\theta^{[i]}$ of a ϕ-searchlight $s^{[i]}$ to be the angular position of the bisector of its field of view,[o] we have the following definition.

Definition 19 (ϕ-Searchlight Critical Angle). An angle ψ is a *critical angle* of a ϕ-searchlight $s^{[i]}$ if $\theta^{[i]}$ passing through ψ implies a change in the set of searchlight critical angles illuminated by $s^{[i]}$'s field of view.

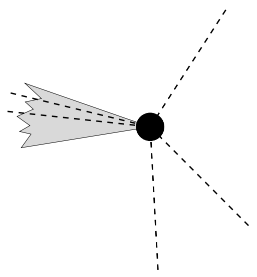

Fig. 15. Supposing the dotted lines are searchlight critical angles, here is shown that a ϕ-searchlight may have multiple searchlight critical angles in its view at any given time. The ϕ-searchlight critical angles are thus defined in terms of the searchlight critical angles as in Definition 19.

Analogous to Definition 14, we can define a critical sequential schedule for ϕ-searchlights to be one in which only one ϕ-searchlight is active at a time and each ϕ-searchlight may only rotate between ϕ-searchlight critical angles. Furthermore, all the Lemmas, Theorems, and Corollaries which we proved for searchlights in Section 3 can be proven analogously for ϕ-searchlights simply by substituting the new definition of critical angle. The proofs are so similar that we omit them and simply state the main (solution space) reduction result.

Corollary 3. *Any instance of the ϕ-Searchlight Scheduling Problem which permits a general search schedule also permits a critical sequential search schedule.*

Just as Corollary 1 led to the design of the complete algorithm for searchlight scheduling in Section 4, Corollary 3 allows a complete algorithm for ϕ-searchlight scheduling.

6. Conclusions

In this article we have shown that the Searchlight Scheduling Problem can be reduced to a path planning problem through an appropriate information graph. The proof was based on an exact cell decomposition of the searchlights' toroidal configuration space. Using the reduction result we designed a complete algorithm for

[o]Taking the bisector is rather arbitrary, we just need a reference angle.

searchlight scheduling. The algorithm is divided into two parts. First, geometric preprocessing is performed in time polynomial in the number of guards and environment vertices. Second, the information graph is searched breadth-first. Our time complexity upper bound for the information graph breadth-first search is exponential in the output size. Although it remains an important open question whether the general Searchlight Scheduling Problem is NP-hard, computed examples demonstrated that the algorithm can be practical for problem instances of useful size, and for which there currently exists no other algorithm. Additionally, we have shown that our complete algorithm for searchlight scheduling can be directly extended to the ϕ-Searchlight Scheduling Problem in which sensors have finite fields of view.

In the future, we hope that NP-hardness of the Searchlight Scheduling Problem can be shown, or else that the computational time complexity bounds for a complete algorithm can be improved. There are also many interesting and unexplored variations of the Searchlight Scheduling Problem. These include minimizing time to clear the environment, evaders with bounded speed, sensor limitations such as limited depth of field, and sensors sweeping a half-plane or cone through three dimensional environments.

Acknowledgments

This material is based upon work supported in part by AFOSR MURI Award F49620-02-1-0325, NSF CMS Award 0626457, and the DoD SMART Fellowship program.

References

1. K. Sugihara, I. Suzuki, and M. Yamashita, “The searchlight scheduling problem,” *SIAM Journal on Computing*, vol. 19, no. 6, pp. 1024–1040, 1990.
2. H. Hattori, S. Nakano, and T. Nishizeki, “Searchlight schedulings for simple polygons,” *Transactions of the Japan Society for Industrial and Applied Mathematics*, vol. 7, no. 3, pp. 265–279, 1997.
3. M. Yamashita, I. Suzuki, and T. Kameda, “Searching a polygonal region by a group of stationary k-searchers,” *Information Processing Letters*, vol. 92, no. 1, pp. 1–8, 2004.
4. B. P. Gerkey, S. Thrun, and G. Gordon, “Visibility-based pursuit-evasion with limited field of view,” *International Journal of Robotics Research*, vol. 25, no. 4, pp. 299–315, 2006.
5. M. Yamashita, H. Umemoto, I. Suzuki, and T. Kameda, “Searching for mobile intruders in a polygonal region by a group of mobile searchers,” *Algorithmica*, vol. 31, no. 2, pp. 208–236, 2001.
6. B. Simov, G. Slutzki, and S. M. LaValle, “Pursuit-evasion using beam detection,” in *IEEE Int. Conf. on Robotics and Automation*, (San Francisco, CA), pp. 1657–1662, Apr. 2000.
7. J. H. Lee, S. M. Park, and K. Y. Chwa, “Simple algorithms for searching a polygon with flashlights,” *Information Processing Letters*, vol. 81, no. 5, pp. 265–270, 2002.
8. J. Urrutia, “Art gallery and illumination problems,” in *Handbook of Computational Geometry* (J. R. Sack and J. Urrutia, eds.), pp. 973–1027, North-Holland, 2000.
9. J. O’Rourke, *Art Gallery Theorems and Algorithms.* Oxford University Press, 1987.

10. T. C. Shermer, “Recent results in art galleries,” *Proceedings of the IEEE*, vol. 80, no. 9, pp. 1384–1399, 1992.
11. U. M. Erdem and S. Sclaroff, “Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements,” *Computer Vision and Image Understanding*, vol. 103, no. 3, pp. 156–169, 2006.
12. A. Ganguli, J. Cortés, and F. Bullo, “Distributed deployment of asynchronous guards in art galleries,” in *American Control Conference*, (Minneapolis, MN), pp. 1416–1421, June 2006.
13. A. Ganguli, J. Cortés, and F. Bullo, “Visibility-based multi-agent deployment in orthogonal environments,” in *American Control Conference*, (New York), pp. 3426–3431, July 2007.
14. K. J. Obermeyer, A. Ganguli, and F. Bullo, “Asynchronous distributed searchlight scheduling,” in *IEEE Conf. on Decision and Control*, (New Orleans, LA), pp. 4863–4868, Dec. 2007.
15. L. J. Guibas, J. C. Latombe, S. M. LaValle, D. Lin, and R. Motwani, “A visibility-based pursuit-evasion problem,” *International Journal of Computational Geometry & Applications*, vol. 9, no. 4-5, pp. 471–493, 1999.
16. S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006. Available at `http://planning.cs.uiuc.edu`.
17. B. Simov, G. Slutzki, and S. M. LaValle, “Clearing a polygon with two 1-searchers,” *International Journal of Computational Geometry & Applications*, 2007. to appear.
18. A. Efrat, L. J. Guibas, D. C. Lin, J. S. B. Mitchell, and T. M. Murali, “Sweeping simple polygons with a chain of guards,” in *ACM-SIAM Symposium on Discrete Algorithms*, (San Francisco, CA), pp. 927–936, Jan. 2000.
19. H. Choset, E. Acar, A. A. Rizzi, and J. Luntz, “Exact cellular decompositions in terms of critical points of Morse functions,” in *IEEE Int. Conf. on Robotics and Automation*, (San Francisco, CA), pp. 2270–2277, Apr. 2000.
20. H. Choset, “Nonsmooth analysis, convex analysis, and their applications to motion planning,” *International Journal of Computational Geometry & Applications*, vol. 9, no. 4-5, pp. 447–469, 1999.
21. S. K. Ghosh, *Visibility Algorithms in the Plane*. Cambridge University Press, 2007.
22. J. O’Rourke, *Computational Geometry in C*. Cambridge University Press, 2000.
23. D. Halperin, “Arrangements,” in *Handbook of Discrete and Computational Geometry* (J. E. Goodman and J. O’Rourke, eds.), pp. 529–562, New York: Chapman and Hall/CRC Press, 2 ed., 2004.
24. S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2 ed., 2003.
25. K. J. Obermeyer, “The VisiLibity library.” `http://www.VisiLibity.org`, 2008. R-1.
26. L. Guibas, D. Salesin, and J. Stolfi, “Epsilon geometry: building robust algorithms from imprecise computations,” in *ACM Symposium on Computational Geometry*, (New York), pp. 208–217, June 1989.
27. M. Segal, “Using tolerances to guarantee valid polyhedral modeling results,” in *Proceedings of SIGGRAPH*, (Dallas, TX), pp. 105–114, Aug. 1990.