Technische Universität Braunschweig

Diploma Thesis

Searching for radio beacons among geometric obstacles with mobile agents

Henning Hasemann

Advisor: Prof. Dr. Sándor P. Fekete

Institute of Operating Systems and Computer Networks

Algorithms Group

Prof. Dr. Sándor P. Fekete

Erklärung

Ich versichere, die vorliegende Arbeit selbstständig und nur unter Benutzung der angegebenen Hilfsmittel angefertigt zu haben.

(Ort, Datum) (Unterschrift)

Zusammenfassung

In dieser Arbeit stellen wir Algorithmen vor, die es einem Agenten ermöglichen, eine Funksignalquelle in einer unbekannten zweidimensionalen Umgebung zu finden. Wir wollen dabei zwei verschiedene Modellklassen betrachten. Die erste Klasse modelliert eine Umgebung die es dem Agenten erlaubt, die Richtung des Ziels zu bestimmen und seine eigenen Rotationen zu verfolgen. Lage und Form der Hindernisse sind dabei dem Agenten nicht bekannt und der Agent hat keine Möglichkeit, seine eigene Position zu bestimmen.

In der zweiten Modelklasse betrachten wir eine realitätsnahe Funksignalquelle und einen Agenten, der in der Lage ist, die Signalstärke an seiner Position in Form eines diskreten Wertes wahrzunehmen. Auch in diesem Fall sollen Lage und Form der Hindernisse dem Agenten nicht bekannt sein. Wir untersuchen dabei verschiedene Ausbreitungsformen des Signals und stellen effiziente Algorithmen zur Lösung des Problems vor.

Abstract

In this work we will present a number af algorithms that make it possible for an agent to find a radio source in an unknown two-dimensional environment. We will consider two different model classes. The first class defines an environment that allows an agent to determine the direction of the radio source and to track its own rotations. Location and shape of the obstacles thereby are not known to the agent and the agent has no means of determining its position.

In the second model class we consider a realistic radio source and an agent that is able to perceive the signal intensity at its position in form of a discrete value. Also in this case location and form of the obstacles shall not be known to the agent. We analyze different signal propagation shapes and present efficient algorithms for solution of the problem.

Contents

List of Algorithms

List of Figures

Chapter 1

Introduction

The problem of finding a radio source with mobile agents among geometric obstacles in an unknown environment has been widely discussed. The nature of the solutions vary substantially with the sensor model that is chosen for the implementing agent. A number of approaches focuses on a model in which the agent knows its exact position relative to the source at all times such as the well known "bug" family of algorithms [8]. Partly they also grant the agent the ability to perceive parts of their environment directly [10, 6, 13]. Others only require knowledge about the direction of or the distance to the radio source [14, 4].

All of these approaches have in common that they use rather simplified models of radio signal distributions or presume an information source other then a radio signal. However real radio signals often do not provide direct angular or distance related information, but still give enough information to find a direct path to the signal source [6, 17]. The idea being that even non-isotropic intensity landscapes can provide a path to the signal source which can be found by following the direction of steepest signal intensity ascent.

However some real world intensity measurement systems such as link quality indicators (LQI, [9]) which are commonly used in wireless sensor networks do only provide discrete signal intensities. In a setup like that a signal intensity ascent may not be measurable without extensive motion and thus the formerly noted approaches can not be applied. Also due to reflection effects there might be multiple maxima of signal intensity at locations other than the radio source [16]. By combining covering techniques with traditional motion planning approaches, we are able to present algorithms which can deal with all of these constraints and still guarantee to find the signal source.

We will start our work by clarifying some preliminaries such as notational conventions, used mathematical concepts and basic algorithms in Chapter 2. Then we will examine two classes of models for the problem in Chapters 3 and 4. In Chapter 3 we will assume that the agent can only use angular input. More precisely, we assume an agent that can track its own rotations and determine the relative direction of the signal source but not its own position or that of the source. We will present a new solution to the problem and compare our findings to the known Angulus algorithm [14]. In Chapter 4 we consider an agent that can only perceive discrete signal intensities but is able to track its own position. The section is divided into subsections that deal with four different types of discrete signal patterns. First we consider circular intensity rings (Section 4.2) which, as we will show, pose a task reducible to the solved problem of finding a signal source when its position is known. In Section 4.3 we will consider discrete signal intensities in general (i.e. with no fixed signal form) and provide algorithms that find a signal source in environments with an almost arbitrary intensity distribution. We provide an optimized variant of that solution in Section 4.4 which can be applied if the environment is known to be star shaped. Section 4.5 discusses how a bound for the length of the agents path can be specified with little more demands on the environment and presents two algorithms that behave efficiently in such environments.

Chapter 2

Preliminaries

2.1 Notation

We utilize prevalent notational conventions for sets. That is, $\mathbb{N} = \{1, 2, \dots\}$ is the set of natural numbers, $\mathbb{N}_0 = \{0, 1, 2, \dots\} = \mathbb{N} \cup \{0\}$ is the set of natural numbers that includes 0. $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$ is the set of integers. $\mathbb{R}$ is the set of real numbers, $\mathbb{R}^+ = \{x \in \mathbb{R} | x > 0\}$ is the set of positive real numbers and $\mathbb{R}_0^+ = \{x \in \mathbb{R} | x \geq 0\}$ is the set of non-negative real numbers. For any set $M = \{m_1, m_2, \dots\}$ we write 2^M in order to refer to the power set of M which is $2^M = \{\emptyset, \{m_1\}, \{m_2\}, \{m_1, m_2\}, \dots\}$.

We use the common notation for sequences i.e. $(a_n)_{n \in \mathbb{N}} = (a_1, a_2, \dots)$ will be handled equivalently to a function $a : \mathbb{N} \to A$ (the codomain A being an arbitrary set), thus we will use the notation $a(n)$ for access to the n'th member. We will also identify tuples with sequences of finite length, so $(e_1, e_2, \dots, e_N)$ for $e_1, \dots, e_N \in E$ defines a sequence $e : \{1, \dots, N\} \to E$ so that $e(n) = e_n$ for all $n \in \{1, \dots, N\}$. We will also allow the construction of new sequences by concatenation of sequences with the same codomain, requiring all but the last sequence to be finite. Syntactically we will denote that with the concatenation operator "$|$". Be $a = (a_1, a_2, \dots, a_N)$, $b = (b_1, b_2, \dots, b_M)$, $c : \mathbb{N} \to A$ sequences with codomain A (i.e. $a_1, \dots, a_N, b_1, \dots, b_M \in A$), then $a|b|c$ denotes the sequence $(a_1, \dots, a_N, b_1, \dots, b_M, c_1, c_2, \dots)$. In this notation an element of the sequences codomain will be treated as a finite sequence of length 1 (e.g. $(1, 2)|3|(4, 5) = (1, 2, 3, 4, 5)$). We will primarily make use of this notation in Section 3 when defining a sequence of binary sequences.

2.2 Topological terms

A subset $M \subseteq \mathbb{R}^n$ is called a topological space if for each point $p \in M$ there is a neighborhood $M \supseteq U_\varepsilon(p) = \{q \in M | \, \|p - q\| < \varepsilon\}$ with $\varepsilon > 0$ which is not empty. In other words, $M \subseteq \mathbb{R}^n$ is open. The set of all neighborhoods $U_\varepsilon(p)$ is called topology on M. A topological space $M \subseteq \mathbb{R}^2$ is called *connected* if there is a path from each point $p_1 \in M$ to each other point $p_2 \in M$ that is contained completely in M. If additionally each path from p_1 to p_2 can be continuously transformed into each other path from p_1 to p_2 we say M is *simply connected* (in a graphic manner, M has no "holes").

A n-dimensional manifold is a topological space that behaves locally like (is homeomorphic to) the euclidean space $\mathbb{R}^n$. However the complete manifold does not need to be homeomorphic to $\mathbb{R}^n$ [12]. For example consider a two-dimensional area like $M_{\text{hole}} = \left\{(x,y) \in \mathbb{R}^2 | 1 < \sqrt{x^2+y^2} < 2\right\}$ which has one hole. Local subsets of M_{hole} have the same properties as $\mathbb{R}^2$, but the whole space does not. E.g. M_{hole} is not simply connected although local subsets of M_{hole} are. A function which converts coordinates between such a local subset of M and the corresponding euclidean space $\mathbb{R}^n$ is called *map*.

A manifold is called *compact* if an infinite number of steps of movement (of constant length) in it eventually reaches every point. As an example, $\mathbb{R}^2$ itself is not compact, but M_{hole} is. A manifold is called *smooth* iff the change of maps between local subsets of the manifold and its corresponding euclidean space is infinitely differentiable. All topological manifolds of dimensions 1, 2 and 3 are smooth[5].

Given a compact, smooth n-dimensional manifold $M \subseteq \mathbb{R}^n$ and a smooth function $f : M \to \mathbb{R}$, we call $p \in M$ a *critical point* of f if $\frac{\partial f}{\partial x_1}(p) = \ldots = \frac{\partial f}{\partial x_n}(p) = 0$. A critical point p is called *non-degenerate* iff its Hessian matrix $H(f)$ is non-singular in p. The number of negative eigenvalues of $H(f)$ at a critical point is called *Morse index* of that point. f is a *Morse function* if all critical points of f are non-degenerate.

> The basic idea of classical Morse theory is that the homotopy (or, in better situations, even the homeomorphism or diffeomorphism) type of the sub-manifold $M_a = \{p \in M \mid f(p) \leq a\}$ changes only at the critical points of f. If there is no critical value in the interval $[a,b]$, then the gradient flow of f provides a diffeomorphism between M_a and M_b [15].

2.3 The mobile agent and its environment

In order to reason about mobile agent motion planning we model the set of positions called *configuration space* $\mathcal{C} \subseteq \mathbb{R}^2$ as a simply connected area. With $\mathcal{C}_{\text{free}} \subseteq \mathcal{C}$ we denote the subset of the configuration space the agent is allowed move freely in. If not further specified we assume that $\mathcal{C}_{\text{free}}$ is bounded by a smooth closed curve which will be treated as an obstacle wall by the following definitions and all presented algorithms.

If the agent is touching the wall of an obstacle it still can move, but obviously not in all directions. We call the set of those points which only allow partial motion $\mathcal{C}_{\text{semi-free}} \subseteq \mathcal{C}$. For the sake of descriptiveness we want to think of the agent as being point-shaped. Note that by adjusting $\mathcal{C}_{\text{free}}$ and $\mathcal{C}_{\text{semi-free}}$, the algorithms presented are applicable to other agent shapes as well, for example using Minkowski sums [7].

In this context we understand an obstacle as an area bounded by a smooth closed curve of finite length in $\mathbb{R}^2$ (called “wall”) whose interior is a subset of $\mathcal{C}_{\text{blocked}} = \mathcal{C} \setminus \{\mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{semi-free}}\}$. For the “outer wall” (the boundary curve of $\mathcal{C}_{\text{free}}$), if existent, it is the outside of the curve which is a subset of $\mathcal{C}_{\text{blocked}}$. We will always assume that there is only a finite number of obstacles and that they are not intersecting or touching each other so that a mobile agent that follows a wall of an obstacle long enough will eventually reach its starting position again.

The signal source will be represented as a 2-dimensional point $p_T \in \mathcal{C}$. The tuple $(\mathcal{C}_{\text{free}}, \mathcal{C}_{\text{semi-free}}, \mathcal{C}_{\text{blocked}}, p_T)$ together with any possible intensity function is called *environment*.

2.4 The Pledge algorithm

The Pledge algorithm solves the problem of escaping from an unknown two-dimensional labyrinth [1]. The only requirement to an executing agent is that it can walk straight facing a certain (arbitrary but fix) direction α_0 in free space and follow obstacle walls until it faces that direction.

Whereas in the latter part it is crucial that the orientation of the agent is not viewed modulo 2π as usual but instead higher and also negative values are considered. E.g. if the agents orientation is $2\pi - \varepsilon$ and the agent executes a 2ε counterclockwise rotation, its new orientation is $2\pi + \varepsilon$ instead of ε.

Algorithm 2.1 Pledge

Notes:

- α_0 is a previously chosen arbitrary but fix angle

```
turn until facing $\alpha_0$
while *true* {
  walk straight until obstacle hit
  turn right and follow wall until facing $\alpha_0$
}
```

Given a finite amount of obstacles with finite circumference, an agent executing 2.1 will eventually leave any region which contains obstacles and “escape” facing direction α_0. Kamphans and Langetepe showed in [11] that the agent does not necessarily need to walk exactly with heading α_0 in free space but can vary the direction as long as the difference in heading at two arbitrary points in free space is at most π.

2.5 The Angulus algorithm

Lumelsky and Tiwari first introduced the Angulus algorithm in [14]. In this section we will briefly explain how the algorithm works, for further input on this topic please see the referenced paper.

Consider the problem of a mobile agent in an unknown, two-dimensional environment that does not know its own position. The agent is equipped with a compass-like device that allows it to detect the direction of the target it is trying to reach. Furthermore, the agent can track its own rotations and follow obstacle walls. Just like for the Pledge algorithm, there is no equivalence of angles which differ by multiples of 2π. Consider the angle β which describes the relative direction of the source (as perceived by the agent) and depends on that agents orientation $\alpha \in \mathbb{R}$. When the agent faces the source directly, $\beta = 0$ holds. If with $\beta = 0$ the agent does a full left turn, it physically faces the source again, but now has an orientation towards the source of $\beta = 2\pi \neq 0$.

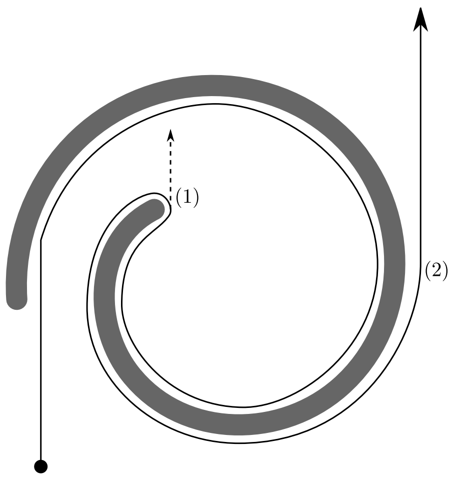


Figure 2.1: Example execution of the Pledge algorithm. Note that at (1) the agents orientation is not α_0 but $\alpha_0 + 2\pi$ which is differentiated so the agent does not leave the obstacle at this point but at (2).

For our description of the algorithm we will also consider the angle $\gamma \in \mathbb{R}$ with $\gamma = \alpha + \beta$ which describes the absolute angle between agent and target. Note that γ does not change when the agent rotates in place.

Algorithm 2.2 Angulus

```
$\gamma_0$ := $\gamma$
while target not reached {
  turn right and follow wall until $\beta = 0$
  if during wall-following $\gamma > \gamma_0 + 2\pi$ {
    continue following wall
      until $(\gamma = \gamma_0 + 2\pi) \wedge (\beta \in [2\pi, 4\pi))$
    $\gamma$ := $\gamma_0$
    turn until $\beta = 2\pi$
  }
  walk straight until obstacle hit
}
```

2.6 Morse decompositions of unknown environments

It is possible to cover an unknown environment efficiently using Morse decompositions by splitting it at critical points. Given that $\mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{semi-free}}$ is a compact, smooth, two-dimensional manifold, and $f : \mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{semi-free}} \to \mathbb{R}$ is a Morse function, f can be visualized as a distance measure from the starting point.

Non-degenerate critical points are exactly those points where the tangent of an obstacle wall is also tangent to a curve which has a constant value under f (we call this curve f-line for short). The f-lines through critical points are used

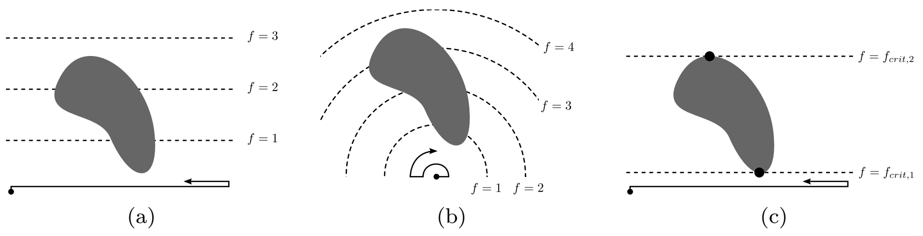


Figure 2.2: a) Covering using a linear pattern (continuous line) and the according Morse function (dashed lines show constant curves under f) b) Covering using a circular pattern and the according Morse function c) Linear covering pattern with curves constant under f (dashed) that share a tangent with points on an obstacle border (black points). These curves are used to split the environment into cells.

to split the environment into *cells* with the property that each cell is completely coverable with a motion pattern that mainly moves along f-lines. After the environment has been split, all “new” cells are added to a list of cells to be covered. During processing of that list, possibly more cells are appended as they are detected. If the agent encounters a dead end situation but still has cells to cover in its list, the agent will move back through already covered cells in order to fulfill that task. Eventually all cells will have been found and covered [2, 3].

Chapter 3

Angular input only

We consider the problem of finding a radio source under the assumption of an agent operating on angular input only. The mobile agent is equipped with a device that provides the direction of the signal source at any time and we assume the agent can track its own rotations. The motions of the agent are restricted to directly head towards the signal source and follow obstacle walls.

This problem definition is related to that of the Pledge algorithm introduced in Section 2.4. However instead of any obstacle-free space the agent has to reach a single point, the signal source. A naive adaption of the Pledge algorithm would always just head at the signal source instead of a fixed direction α_0 and circumvent obstacles analog to the usual Pledge manner: Turn right and follow the obstacle wall until facing the signal source again, take care to distinguish 0 and 2π.

Figure 3.1a shows an example where this approach is successful. If all obstacles and the signal source can be divided by a straight line as shown, the directions of two free space motions do not differ by more than π. It has been shown that an agent executing the Pledge algorithm can be subject to an absolute "measurement error" regarding its own orientation about up to π and still reach its goal [11].

Now consider the case in Figure 3.1b where the signal source is on the inside of a spiral-shaped obstacle. The agent executes the same Pledge-like algorithm which results in an endless loop. This is solved by the Angulus algorithm (introduced in Section 2.5) by detecting the circling and not leaving the obstacle during the second circling until the agent made it into the spiral [14]. With our approach on the other hand the agent will still leave the obstacle at that position but alter the rule "turn right and follow obstacle" into "turn left and follow obstacle". In general our agent tries to find a sequence of these rules (hence called *turn plan*) which will let it reach the signal source eventually.

3.1 Model

We consider the problem of finding a radio source under the assumption of an agent operating on angular input only. More precisely the agent has the following capabilities:

(1) The agent will detect when it has physical contact to the radio source.

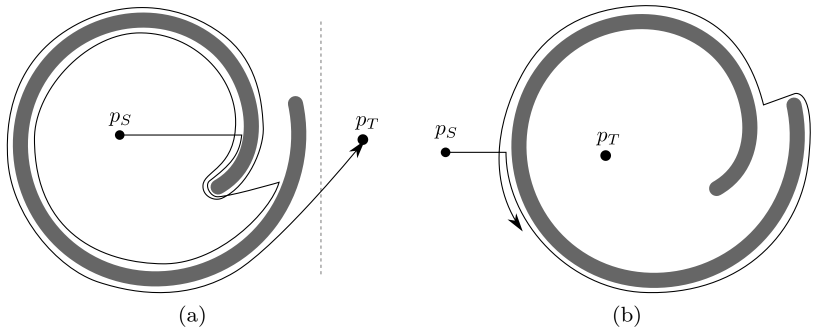


Figure 3.1: a) A naive Pledge-like solution can be successful if the signal source and all obstacles are dividable by a straight line. b) If that is not the case for example because the signal source is enclosed by a spiral-like obstacle, the naive approach can lead to infinite loops.

(2) During all its motions the agent is able to track its cumulative turning angle in a variable $\alpha \in \mathbb{R}$. In our case we do not apply the otherwise usual equivalence of a turning angle α with the angles $\alpha + 2\pi k$ for all $k \in \mathbb{Z}$. I.e., in this model 0, 2π and -2π would be three different angles.

(3) The agent can measure the relative direction of the source $\beta \in \mathbb{R}$ (see Figure 3.2). Like α, no implicit modulus 2π operation is applied.

(4) The agent can turn until it faces the signal source ($\beta = 0$).

(5) When the agent has just touched a wall it can turn either left or right and follow the wall. Whenever it is oriented in a way so that $\beta = 0$, i.e. the agent looks at the signal source, it can leave the current obstacle in a straight line until it touches either the radio source or another wall.

In order to detect circling around the source, the agent tracks the direction β of the source during all motions. However in determining if the source has been circled, the direction the agent itself is facing (α) should not be relevant, thus instead of observing β directly, we usually observe $\gamma = \alpha + \beta$ as depicted in Figure 3.2.

Then γ is the angular position of the source relative to the agent irrespective of which direction the agent is facing. When calculating this angle at two different points in time leading to two values γ_0 and γ_1, the source has been circled $\frac{\lfloor \gamma_0 - \gamma_1 \rfloor}{2\pi}$ times between the measurements. As noted in Section 2.3 we assume that in each environment there is only a finite number of obstacles, each having a smooth closed curve of finite length as boundary. We also assume that $\mathcal{C}_{\text{free}}$ is *not* bounded by a smooth closed curve so the environment is not shaped like a courtyard but rather like an open field with limited obstacles.

3.2 Orientation based motion planning

When the agent circles the signal source it has to alter its turn plan (the sequence of left/right turn rules applied after hitting a wall) in a smart way to be able

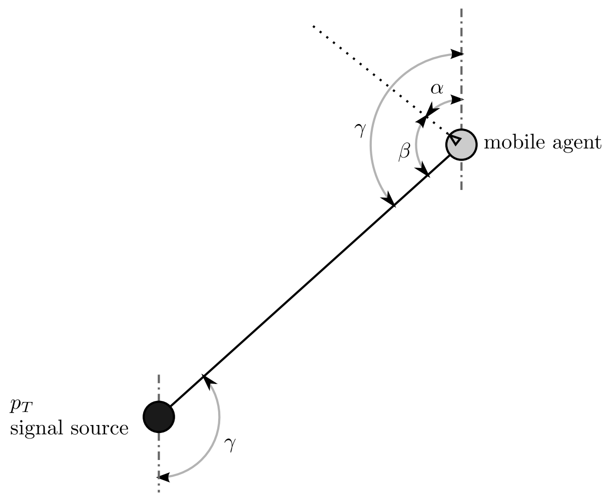


Figure 3.2: Angular input to the agent

to bypass the obstacle(s) it circles and get nearer to the source. In order to achieve this, we introduce ϕ, a sequence of binary sequences that will be used to explore different turn plans.

Definition 3.2.1. We define $\phi : \mathbb{N} \to (\mathbb{N} \to \{0,1\})$, a sequence of binary sequences by

$$\begin{aligned} \phi(1) &= 000\ldots \\ \phi(n) &= b_m(n - 2^m - 1)|1|\phi_1 \text{ with } m = \lceil \log_2 n \rceil - 1 \end{aligned}$$

The notation $a|b$ means concatenation of tuples and sequences in this context. We define $b_m : \mathbb{N}_0 \to \{0,1\}^m$ where $b_m(v)$ is the m-tuple of bits that is the binary representation of the value v beginning with the most significant bit (MSB) (e.g. $b_3(6) = (1,1,0)$). As a special case $b_0(v)$ shall always be the tuple of length 0 regardless of the value of v.

For the n'th member sequence of ϕ we will write $\phi(n)$.

Corollary 3.2.2.

$$\phi(n) = \phi(m) \Leftrightarrow m = n$$

holds for all $n, m \in \mathbb{N}$.

Regarding the members of any $\phi(m)$ we will identify the value 0 with the rule “turn right and follow wall” (R) and 1 with “turn left and follow wall” (L) respectively. These rules will later on be applied whenever the agent touches an obstacle. For the sake of mathematical argumentation about counting- and similar properties we will continue to use the numerical notation most of the time though.

Having defined a sequence of turn plans we present Algorithm 3.1 which applies these plans trying out a new one whenever it completed a circle around the signal source. An agent executing the algorithm always heads towards the signal source in free space and follows obstacle borders until it faces the source

Seq.	Bit string	Seq.	Bit string
$\phi(1)$	$\texttt{000000}\ldots$	$\phi(9)$	$\underline{\texttt{000}}\texttt{100}\ldots$
$\phi(2)$	$\texttt{100000}\ldots$	$\phi(10)$	$\underline{\texttt{001}}\texttt{100}\ldots$
$\phi(3)$	$\underline{\texttt{0}}\texttt{10000}\ldots$	$\phi(11)$	$\underline{\texttt{010}}\texttt{100}\ldots$
$\phi(4)$	$\underline{\texttt{1}}\texttt{10000}\ldots$	$\phi(12)$	$\underline{\texttt{011}}\texttt{100}\ldots$
$\phi(5)$	$\underline{\texttt{00}}\texttt{1000}\ldots$	$\phi(13)$	$\underline{\texttt{100}}\texttt{100}\ldots$
$\phi(6)$	$\underline{\texttt{01}}\texttt{1000}\ldots$	$\phi(14)$	$\underline{\texttt{101}}\texttt{100}\ldots$
$\phi(7)$	$\underline{\texttt{10}}\texttt{1000}\ldots$	$\phi(15)$	$\underline{\texttt{110}}\texttt{100}\ldots$
$\phi(8)$	$\underline{\texttt{11}}\texttt{1000}\ldots$	$\phi(16)$	$\underline{\texttt{111}}\texttt{100}\ldots$

Table 3.1: The first 16 member sequences of ϕ represented as bit strings. Underlined is the component $b_m(v)$ which is a counting sequence of increasing length. $b_m(v)$ is always followed by a `1` and an infinite number of `0`s except for $\phi(1)$ and $\phi(2)$ where $b_m(v)$ has length 0.

again. Thereby the agent takes care not to identify angles that have a difference of $2\pi n$ in order to avoid "simple" loops (just like the Pledge algorithm). Whenever the agent notices it circled the signal source, it will advance to the next turn plan provided by $\phi(i)$.

We will now introduce a series of utility lemmas and definitions that lay some ground work for analyzing the presented algorithm. We start with some general observations about agents that circle obstacles. More precisely, we show that agents never circle an obstacle completely without facing the signal source at least once. We will use this later on to prove that each obstacle will be left eventually.

Lemma 3.2.3. *Be $P : [0, t_{max}) \to \mathbb{R}^2$ with $t_{max} \in \mathbb{R}^+$ a smooth Jordan curve of finite length and $p_T \in \mathbb{R}^2$ a point outside of this curve.*

Further be $g_1, g_2 : \mathbb{R} \to \mathbb{R}^2$ two rays starting in p_T which intersect P exactly in one point each. We call these points $P(t_1)$ and $P(t_2)$ respectively. Then the following holds:

(1) Both g_1 and g_2 are tangents of P.

(2) An agent moving along P in increasing t direction (and keeping its orientation adjusted to the gradient of P) will look directly at p_T in $P(t_1)$ and directly away from p_T in $P(t_2)$ or vice versa.

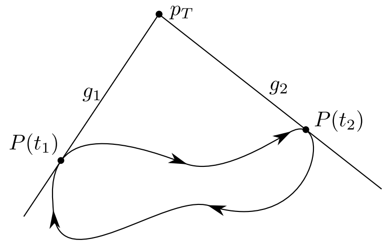


Figure 3.3: Path tangents

Algorithm 3.1 Orientation based motion planning (OBMoP)

Preconditions

- The agent is located in $\mathcal{C}_{\text{free}}$
- $\phi(n)$ as given in Definition 3.2.1

```
i := 1
j := 1
rotate until $\beta = 0$
$\gamma_0$ := $\gamma$
while source not reached {
  move straight until obstacle hit
  if $(\phi(i))\,(j) = 0$ {
    rotate right and follow wall until $\beta = 0$
  }
  else {
    rotate left and follow wall until $\beta = 0$
  }
  $j$ := $j + 1$
  if during wall-following $\gamma \bmod 2\pi = \gamma_0$ {
    $j$ := 1
    $i$ := $i + 1$
  }
}
```

Proof. If any of g_1, g_2 would not be tangent of P, there would be points in P on both sides of the respective ray. Since P is a closed curve that does not circle p_T, that ray would have to intersect P twice and that would contradict the construction of the ray $\lightning$. Thus, g_1 and g_2 are both tangents to P and (1) is fulfilled, see Figure 3.3.

Since P does not cross itself, the "increasing t" direction points at p_T in $P(t_1)$ and away from p_T in $P(t_1)$ (see figure), so (2) holds true as well. $\square$

The central point in the later proof for convergence of our algorithm will be the claim that it does not lead the agent into an infinite loop. Before we can really claim that we need to define the term loop in this context:

Definition 3.2.4. Be $P : \mathcal{T} \to \mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{semi-free}}$ with $\mathcal{T} \subseteq \mathbb{R}_0^+$ and $\mathcal{T}$ connected a path of an agent executing a source finding algorithm. We say P contains a loop (u, v) if both of the following hold

- $u, v \in \mathcal{T}$ and $u < v$.
- The codomain of the interval $[u, v]$ in P, written as $P([u, v])$ is a (closed) curve with $P(u) = P(v)$.

We define $\mathcal{L}_P \subseteq \mathcal{T} \times \mathcal{T}$ as the set of all pairs u, v that fulfill the properties stated above. Further be $\mathcal{L}_P^1 \subseteq \mathcal{L}_P$ the set of all pairs u, v that fulfill the properties stated above and for which $P([u, v])$ is simple (i.e. $P([u, v])$ does not cross itself but is a Jordan curve). We call $\mathcal{L}_P^1$ the set of minimal loops in P. A loop $(u, v) \in \mathcal{L}_P$ is said to circle the source k times iff $|\gamma_v - \gamma_u| = 2\pi k$ for $k \in \mathbb{Z}\backslash\{0\}$. Note that for a loop that does not circle the source there still might be a $\bar{v} \in (u, v)$ so that $|\gamma_{\bar{v}} - \gamma_u| = 2\pi k$ for a $k \in \mathbb{Z}\backslash\{0\}$. This is the case when a loop describes a path around the source but "comes back".

If an agent executing Algorithm 3.1 would loop infinitely, it could either circle the source or not. The first case is what we avoid by the choice and application of ϕ. We will now prove that an agent can not loop forever without circling the source, so we can guarantee that if there is a potentially endless loop we have a chance to alter the turn plan each lap until the agent can "escape" towards the signal source.

Lemma 3.2.5. *Be $P : \mathbb{R}_0^+ \to \mathcal{C}_{free} \cup \mathcal{C}_{semi\text{-}free}$ the path of an agent executing Algorithm 3.1 that contains at least one loop. Be $(u, v) \in \mathcal{L}_P$ a loop in P that does not cross the γ_0 line. Then (u, v) is not an endless loop, i.e. there is a $t_{leave} \in \mathbb{R}$ with $t_{leave} > v$ so that $P([t_{leave}, \infty)) \cap P([u, v]) = \emptyset$.*

Proof. We demonstrate that in this special case Algorithm 3.1 behaves equivalently to the Pledge algorithm which is known to inhibit endless loops. We do this by applying a polar transformation with the radio source as center to all obstacle forms and the agents path. Note

(1) Since the γ_0 line is never crossed, the fact that the polar space exhibits a "wrap-around" behavior at γ_0 is not relevant. More precisely, the polar space is homeomorphic to a cylinder, however the agents path is completely contained in a simply closed two-dimensional surface.

(2) A segment of a path describes a loop in the "normal" space if and only if it describes a loop in the polar space.

(3) Since $\gamma = \gamma_0$ does not occur during the loop, the agents remaining turn plan will eventually become $00000\ldots$ so the agent will eventually default to always turn right and follow the wall whenever it touches an obstacle.

(4) The case of having $\beta = 0$ (the agent is facing the source) in the "normal" space is equivalent to having $\alpha = \alpha_0$ in the polar space (the agent facing a certain direction).

(5) Moving straight towards the source in normal space is equivalent to moving in the α_0 direction in the polar space.

Considering (3) after a finite number of jumps the application of these observations to Algorithm 3.1 delivers a direct description of the Pledge algorithm. Since the pledge algorithm guarantees to eventually leave any group of obstacles and thus any loop, there can be no endless loops either in the polar space nor in its transformation preimage the "normal" space. □

The basic idea of $\phi(n)$ is to try out all possible left/right rule sequences in a way that can not lead to loops. However we did not state yet in which cases such a turn plan is applicable at all and when a correct sequence exists.

Lemma 3.2.6. *Given an environment in which the signal source is reachable from all points in $\mathcal{C}_{free} \cup \mathcal{C}_{semi\text{-}free}$ and an agent that exhibits the following behavior:*

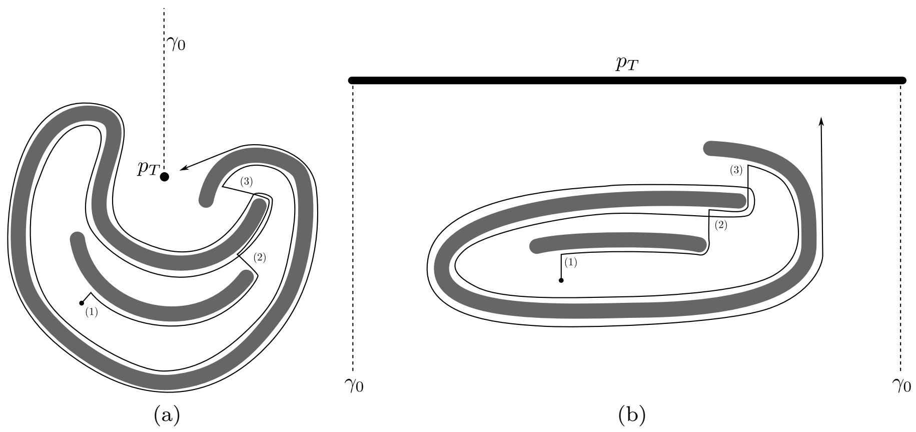


Figure 3.4: a) A path of an agent executing Algorithm 3.1 which does not cross the γ_0 line. b) The same environment and path after a polar transformation with the radio source as center. The behavior of the agent in the polar space is identical to the Pledge algorithm.

- *The agent measures its cumulative relative angle to the signal source $\beta \in \mathbb{R}$. The otherwise usual equivalence of a turning angle β with the angles $\beta + 2\pi k$ for all $k \in \mathbb{Z}\backslash\{0\}$ is not applied ($\beta = 2\pi$ is a different case then $\beta = 0$).*
- *When in free space the agent will move straight towards the source until it touches an obstacle.*
- *After arriving at an obstacle the agent will turn either right or left according to a sequence $a : \mathbb{N} \to \{R, L\}$ so that its n'th turn will be determined by $a(n)$. Then the agent will follow the obstacle until $\beta = 0$ holds and then drive straight towards the source.*

For each possible starting point $p_S \in \left(\mathcal{C}_{free} \cup \mathcal{C}_{semi\text{-}free}\right)$, there is a sequence $a : \mathbb{N} \to \{R, L\}$ so that the agent will reach the signal source after having traveled a path of finite length.

Proof. We prove by induction over the number of obstacles in the environment. □

Basis If there are no obstacles in the environment, the agent will obviously reach the environment with every finite sequence $a : \mathbb{N} \to \{R, L\}$ since it never touches an obstacle but only moves straight to the source.

Step Assume, there is already a finite sequence $a : \mathbb{N} \to \{R, L\}$ so that the agent starting at p_S will reach the source using that sequence. After adding a new obstacle in a way that the radio source still is reachable from all points in the new free and semi-free configuration space, there exists a (possibly different) finite sequence $a' : \mathbb{N} \to \{R, L\}$ so that the source will be reached from p_S when a' is applied.

This can be shown as follows: If the new obstacle (O for short) does

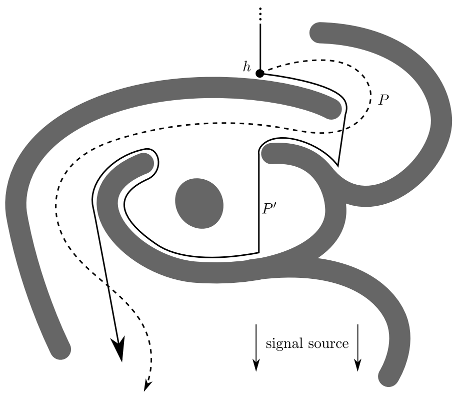


Figure 3.5: If there is a path P in $\mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{semi-free}}$ that leads from the outside of an obstacle to its inside then there exists a path P' that with the same starting point which also ends on the inside of the obstacle and consists solely of motions along obstacle boundaries and straight toward the signal source. Note that P and P' are not necessarily homeomorphic to each other.

not intersect the former path of the agent from p_S to the radio source, then $a' = a$ will certainly fulfill the condition. Else a can be split in two sub-sequences a_{before} and a_{after} so that $a_{\text{before}} | a_{\text{after}} = a$ and after applying a_{before} the agent will touch O for the first time at point $h \in \mathcal{C}_{\text{semi-free}}$. There is a path P from h to the inside of O (else the source would not be reachable anymore). As P is a path from the outside of an obstacle to its inside, there is also a similar path that leads to the obstacles inside using only motions as described in the lemma as illustrated by Figure 3.5, which is constructed by "projecting" P to the nearest obstacle in direction towards the signal source.[1]

After overcoming O the agent might not be on a point where executing a_{after} will lead to the radio source, but in any case the agent will be nearer to the source. However since the source was reachable before insertion of O from all points in the free configuration space and we have shown that O is passable from every point h towards the source there must also be a sequence a'_{after} that leads the agent from O to the radio source.

Imagine an an agent starting from the γ_0-line at point p_1. It is possible that a certain turn plan $\phi(x)$ leads the agent to a point $p_2 \neq p_1$ with $\gamma_{p_2} \bmod 2\pi = \gamma_{p_1} \bmod 2\pi = \gamma_0$. At p_2 the agent executes the turn plan $\phi(x+1)$ so that it arrives at p_1 again, so the agent walks a loop. This is fine per se, however we have to ensure that this does not go on forever (i.e., $\phi(x+2)$ leads again to p_2 and so on). Note that the agent only executes a finite number of turns from each plan, so even though $\phi(x) \neq \phi(x+2) \neq \phi(x+4) \neq \dots$ holds, they may in general still have a common prefix so that the agent will loop forever.

In the following lemma we show that ϕ as given in Definition 3.2.1 can not repeat prefixes forever this way, we call this property *non linear prefix repetition.* First however, we need to make two important general observations

[1]Actually the rules for this transformation are a little more complex than a simple projection, the procedure should be intuitively clear though.

about counting sequences.

Observation 3.2.7 (Repetition in counting sequences)**.** *Be $m, p \in \mathbb{N}$ with $p < m$. Then a binary counting sequence with m bits repeats each prefix of length p 2^{m-p} times in a row.*

For instance, consider the sequence $c_3 = (000, 001, 010, 011, 100, 101, 110, 111)$. The prefixes of length one are $(0, 0, 0, 0, 1, 1, 1, 1)$ each of which repeats $2^{3-1} = 4$ times. The prefixes of length two are $(00, 00, 01, 01, 10, 10, 11, 11)$, each of which repeats $2^{3-2} = 2$ times.

Observation 3.2.8 (Counting sequences in ϕ)**.** *Given a value $q \in \mathbb{N}$ we define $n_0 = 2^q + 1$, the tuple $(\phi(n_0), \phi(n_0 + 1), \dots, \phi(2n_0 - 2))$ has the binary counting sequence in $\lceil \log_2 n_0 \rceil - 1$ bits as prefix.*

E.g., $(\phi(5), \phi(6), \phi(7), \phi(8)) = (0010\dots, 0110\dots, 1010\dots, 1110\dots)$ has the two bit counting sequence $(00, 01, 10, 11)$ as prefix.

Lemma 3.2.9 (Non linear prefix repetition)**.** *Be $a : \mathbb{N} \to \{0, 1\}$ a binary sequence. Then for any $s, d, k \in \mathbb{N}$ there is an $i \in \mathbb{N}_0$ so that at least the first k members of a and $\phi(s + id)$ are identical.*

Proof. Be $n_0 = 2^{k+\lceil \log_2 d \rceil} + 1$. If $n_0 + d - 1 < s$, we choose a higher value for k so that this is not the case anymore. Then more bits than claimed will be identical which is no contradiction to the original postulation. Note that $n_0 \geq 3$ holds by definition.

Consider the tuple $(\phi(n_0), \phi(n_0 + 1), \dots, \phi(n_0 + d - 1))$. According to Observation 3.2.8, this tuple has a binary counting sequence in $\lceil \log_2 n_0 \rceil - 1 = \left\lceil \log_2 \left(2^{k+\lceil \log_2 d \rceil} + 1\right) \right\rceil - 1 = k + \lceil \log_2 d \rceil$ bits as prefix (note that $2n_0 - 2 \geq n_0 + d - 1$ holds). According to Observation 3.2.7, the k long prefixes repeat $2^{(k+\lceil \log_2 d \rceil)-k} = d$ times in a row in this tuple. Since $\phi(s + id)$ contains every d'th member-sequence, it must eventually take every possible binary sequence of length k as a value.

Note that in general there might be values $n_0' \in \mathbb{N}$ with $n_0' = s + id < n_0$ so that the first k members of $\phi(n_0')$ equal the first k members of any given sequence a. □

Using that property we can now show that with the proposed algorithm the agent will not engage in loops that repeatedly circle the signal source.

Lemma 3.2.10. *Be P the path of an agent executing Algorithm 3.1 that contains at least one loop $(u, v) \in \mathcal{L}_P$ which crosses the γ_0 line. Then (u, v) is not an endless loop, i.e. there is a $t_{leave} \in \mathbb{R}$ with $t_{leave} > v$ so that $P([t_{leave}, \infty)) \cap P([u, v]) = \emptyset$.*

Proof. Within the time interval $[u, v]$, the agent circles the source $N \in \mathbb{N}^+$ times, i.e. there are time indices $u \leq t_1 < \dots < t_N < v$ with $\gamma_{t_1} \bmod 2\pi = \gamma_{t_2} \bmod 2\pi = \dots = \gamma_{t_N} \bmod 2\pi = \gamma_0 \bmod 2\pi$. Since the radio source is reachable, there must be at least one free space motion after which - with an appropriate turn plan - the agent will take a different path than before. This path may already "free" the agent from the former loop, or it may change the loop possibly adding new crossings with the γ_0 line and additional free space motions. Since the number of leave points (points where $\beta = 0$ could possibly hold) is finite, so is the maximum size and number of possible variants of the loop. For one of those variants there must be a leave point though for which one of the decisions “turn right and follow wall” (0) , “turn left and follow wall” (1) will lead to leaving the loop see Figure 3.6.

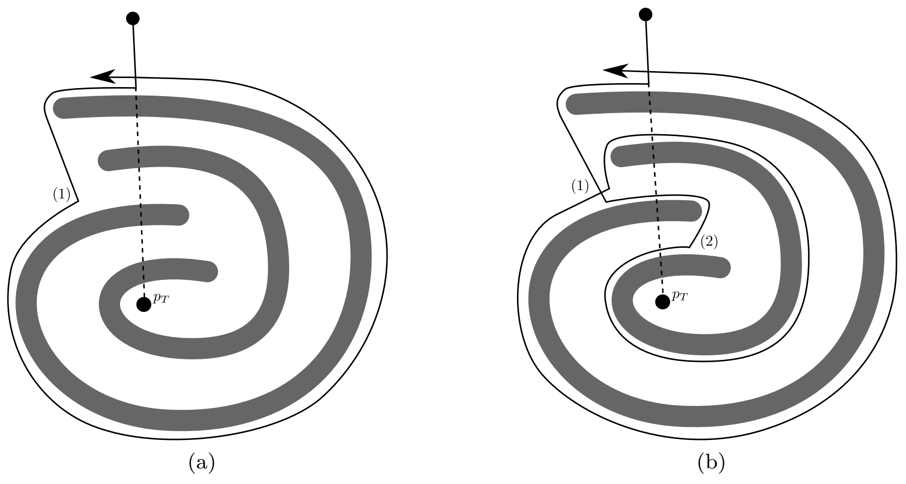


Figure 3.6: a) A loop which can not be left just by correct choice at (1) b) A variant of the loop which contains a leaving possibility at (2)

Be t_L the last time index so that $\gamma_{t_L} \bmod 2\pi = \gamma_0 \bmod 2\pi$ before that leave point. Then there exists a finite sequence a of turning decisions so that if it is applied after t_L it will lead the agent out of the loop. Be $i_L \in \mathbb{N}$ the value of i (index to ϕ in Algorithm 3.1) when the agent first encounters $P(t_L)$. On the j'th visit of $P(t_L)$ the agent will use the sequence $\phi(i_L + jd)$ to determine its prospective path. Due to Lemma 3.2.9 we know there is a $j \in \mathbb{N}$ so that the agent will actually use the sequence a at some point, so it will eventually leave the loop. □

Theorem 3.2.11. *An agent executing Algorithm 3.1 in an environment with reachable signal source will reach the signal source after traveling a path of finite length.*

Proof. Since all free space motions of the agent are aligned towards the radio source this is the case when both the following conditions hold true:

(1) The agent will eventually leave any obstacle so it will move towards the source.

(2) There is no endless loop possible so for each free space motion towards the source there will be a later free space movement which will end nearer to the source than the previous one did. (Naturally that does not have to hold true for the last free space movement)

From Lemma 3.2.3 follows that for each obstacle there is a point so that $\beta = 0$ holds during the first circling of the obstacle, so each obstacle is left eventually, (1) holds.

Due to Lemmas 3.2.5 and 3.2.10, (2) holds too so an agent executing Algorithm 3.1 will eventually reach the signal source. □

Theorem 3.2.12. *Consider an agent executing Algorithm 3.1. For each (arbitrary large) $n \in \mathbb{N}$ and each (arbitrary small) $\varepsilon \in \mathbb{R}^+$ there is*

(1) an obstacle O_S so that the agent travels a path of length $|P| \geq \|p_T - p_S\| + n\,|\partial O_S| - \varepsilon$ before reaching the source. P does not circle the source.

(2) an obstacle O_T so that the agent circles obstacle and signal source at least 2^{n+1} times, traveling a path length of $|P| \geq \|p_T - p_S\| + 2^n\,|\partial O_T| - \varepsilon$ before reaching the source.

Proof.

(1) Analog to the proof of the unboundedness of the Angulus algorithm [14] we prove the same property for Algorithm 3.1, however we only consider the one-obstacle case.

As can be seen in Figure 3.7, the sum of the free space motions is certainly

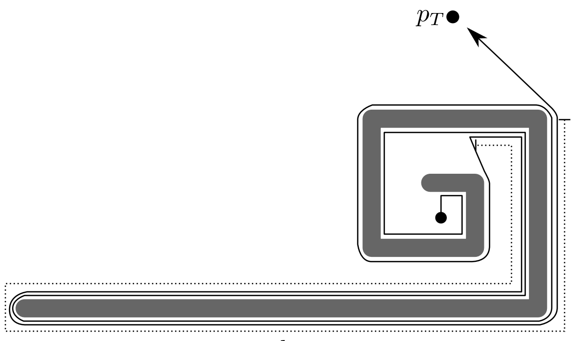


Figure 3.7: An environment that enforces unbounded behavior for Algorithm 3.1 as well as for the Angulus algorithm.

greater or equal to $\|p_T - p_S\|$. By adjusting the number of windings of the spiral around p_S, an arbitrary high number n of traversals of the segment l can be enforced. Other segments of O_S are traveled $n-1$ or fewer times, so by choosing the length of l and/or the "spiral part" so that $|\partial O_S| - l \leq \frac{\varepsilon}{n}$, we see that

$$\begin{aligned} |P| &\geq \|p_T - p_S\| + nl \\ &\geq \|p_T - p_S\| + n\left(|\partial O_S| - \frac{\varepsilon}{n}\right) \\ &= \|p_T - p_S\| + n\,|\partial O_S| - \varepsilon \end{aligned}$$

holds.

(2) We construct an obstacle like shown in Figure 3.8 with n steps. If the agent starts from the γ_0 line in counter-clockwise direction, the sequence for traversal of the obstacle is $a_{\text{ccw}} = 00\ldots01 = 0^n1$. A sequence which does not have a as prefix but contains a 1 in its first n members will make the agent turn around and circle the obstacle clockwise in which case the sequence for traversal is $a_{\text{CW}} = 00\ldots01 = 0^{n+1}1$. In this case if the agent tries a sequence which does contain a 1 in its first $n+1$ members it will

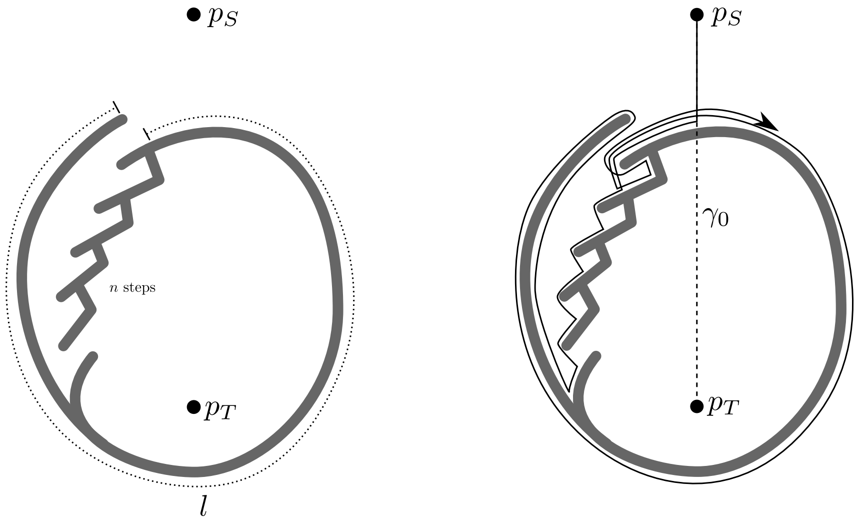


Figure 3.8: A malicious, source-circling environment for Algorithm 3.1. Left: Environment setup, the dotted line depicts the distance l. Right: Partial execution path of the algorithm in this environment. The dashed line represents places where $\gamma \bmod 2\pi = \gamma_0$ holds.

continue to circle clockwise.
The first sequence that has a_{CW} as prefix is $\phi(2^{n+1}+1)$. Note that all sequences before that one contain a 1 except for $\phi(1) = 00\ldots$, so for $n > 1$ the agent is surely circling clockwise directly before traversal of the obstacle. During its clockwise circlings around the obstacle, the agent traverses l at least 2^{n+1} times. By choosing l so that $\frac{1}{2}\left|\partial O_T\right| - l \leq \frac{\varepsilon}{2^{n+1}}$ holds[2] we got

$$\begin{aligned}|P| &\geq \|p_T - p_S\| + 2^{n+1} l \\ &\geq \|p_T - p_S\| + 2^{n+1}\left(\frac{1}{2}\left|\partial O_T\right| - \frac{\varepsilon}{2^{n+1}}\right) \\ &= \|p_T - p_S\| + 2^{n}\left|\partial O_T\right| - \varepsilon\end{aligned}$$

□

3.3 Comparison to the Angulus algorithm

In the previous section we have shown that an agent executing Algorithm 3.1 can be forced to travel arbitrary long paths without circling the signal source. Lumelsky and Tiwari have shown that the same is true for Angulus. Also both algorithms can be forced into generation of long paths while having the agent circle the signal source repeatedly. However the environments that lead to those cases are different for the two algorithms. As we just constructed such an environment for Algorithm 3.1, we will now investigate that case for Angulus.

[2]Note that we can only consider $\frac{1}{2}\left|\partial O_S\right|$ instead of $\left|\partial O_S\right|$ because only the outside of the obstacle is traveled at all.

After that we will compare the properties of the environments in order to say something about the difference between the algorithms.

Theorem 3.3.1. *Consider an agent executing the Angulus algorithm [14]. For each (arbitrary large) $n \in \mathbb{N}$ and each (arbitrary small) $\varepsilon \in \mathbb{R}^+$ there is an obstacle O_T so that the agent circles obstacle and signal source at least $\frac{1}{2}\left(n^2 + n\right)$ times, traveling a path of length $|P| \geq \frac{1}{2}\left\|p_T - p_S\right\| + \frac{1}{6} n \left|\partial O_T\right| - \varepsilon$ before reaching the source.*

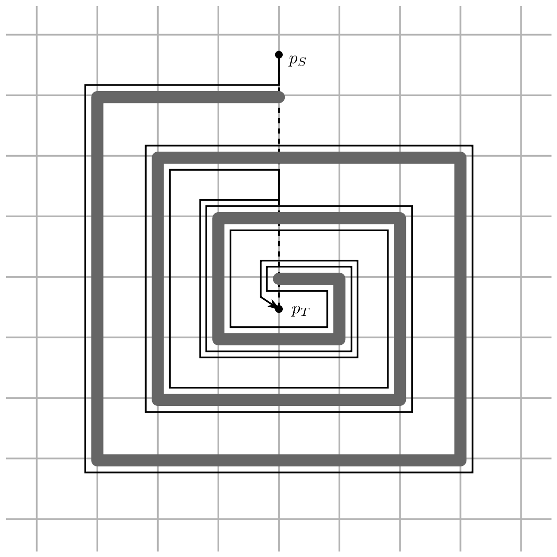


Figure 3.9: Malicious environment for the Angulus algorithm. The dashed line and the gray squares can be used to identify the spiral segments. For example the innermost spiral segment is 7 units long, the next one $7 + 8 = 15$, then $7 + 2 \cdot 8 = 23$ and so on.

Proof. We construct the obstacle as shown in figure 3.9 with n spiral segments, where each segment describes a whole turn around the signal source and the i'th segment has length $7 + 8(i - 1)$. The obstacle is arbitrary thin so the path lengths on the inside and outside of a spiral segment are arbitrary near to the segment length (depending on ε).

In the first phase, Angulus walks the complete "outside spiral" of the obstacle. When the executing agent physically faces the signal source however, it does not leave the obstacle, but continues to follow the wall, as it already circled the source n times, so $\gamma = 2\pi n \geq 2\pi$ holds. The algorithm states that in that case the agent will not leave the obstacle wall until $(\gamma = \gamma_0 + 2\pi) \wedge (\beta \in [2\pi, 4\pi))$ holds, which first is the case at the beginning of the second outermost spiral.

So the agent walks "back" the spiral length minus the outermost segment length. Then the agent executes a free space motion "jumping over" to the beginning of the third outermost spiral segment (leaving out the two outermost segments).

The process repeats until the leave point is in the innermost spiral segment and the agent reaches the source. The number of signal source circlings (when counting both directions) is thus

$$\begin{aligned} & \sum_{i=1}^{n}(n-i+1) \\ = \;& n^2 - \frac{1}{2}n(n+1) + n \\ = \;& \frac{1}{2}\left(n^2+n\right) \end{aligned}$$

The agent travels the outermost segment length once, the second outermost segment length twice and so on, so we can calculate the length l_{O_T} of the path the agent travels along obstacle walls. Note that in this and the following equations we will use ε_x-terms to represent potential effects of the obstacle "thickness" (a thin obstacle results in small values for ε's).

$$\begin{aligned} l_{O_T} &= \sum_{i=1}^{n}(i(7+8(n-i)) - \varepsilon_1 \\ &= (7+8n)\sum_{i=1}^{n} i - 8\sum_{i=1}^{n} i^2 - \varepsilon_1 \\ &= (7+8n)\frac{n(n+1)}{2} - \frac{4n(n+1)(2n+1)}{3} - \varepsilon_1 \\ &= \frac{4}{3}n^3 + \frac{7}{2}n^2 + \frac{13}{6}n - \varepsilon_1 \end{aligned}$$

The obstacle perimeter is nearly two times the spiral length:

$$\begin{aligned} |\partial O_T| &= 2\sum_{i=1}^{n}(7+8(i-1)) + \varepsilon_2 \\ &= 16\sum_{i=1}^{n} i - 2n + \varepsilon_2 \\ &= 8n(n+1) - 2n + \varepsilon_2 \\ &= 8n^2 + 6n + \varepsilon_2 \end{aligned}$$

We set the traveled distance in relation to the length of the obstacle boundary:

$$\begin{aligned} \frac{l_{O_T}}{|\partial O_T|} &= \frac{\frac{4}{3}n^3 + \frac{7}{2}n^2 + \frac{13}{6}n - \varepsilon_1}{8n^2 + 6n + \varepsilon_2} \\ &= \frac{1}{6}n + \frac{7}{192n+144} + \frac{5}{16} - \varepsilon_3 \end{aligned}$$

In addition to the wall-following the agent executes some free space motions. We can see in Figure 3.9 that these motions occur once for every second spiral segment. As the spiral segments divide the line between the agents starting point and the signal source into equal pieces, the length of the agents free space motions is $\frac{1}{2}\|p_T - p_S\| - \varepsilon_4$.

The total path length of the agent is thus

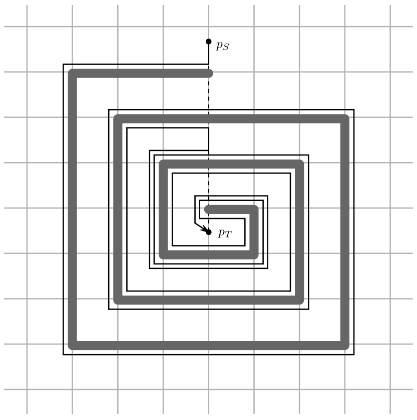
$$\begin{aligned} |P| &= \frac{1}{2}\|p_T - p_S\| + \left(\frac{1}{6}n + \frac{7}{192n+144} + \frac{5}{16}\right)|\partial O_T| - \varepsilon \\ &\geq \frac{1}{2}\|p_T - p_S\| + \frac{1}{6}n\,|\partial O_T| - \varepsilon \end{aligned}$$

□

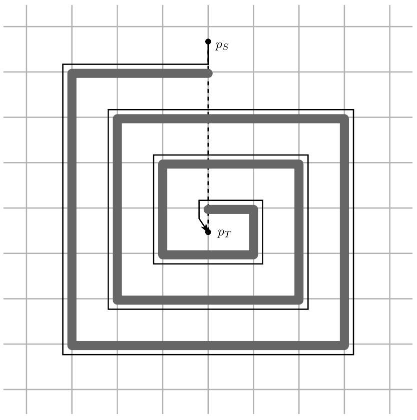
Note that it is certainly possible to create environments which enforce even longer paths for a given number of spiral segments, for example by using (non-rectangular) spirals with a lower “outer segment length” to “inner segment length” ratio (e.g. Fermat’s spiral). We showed however that the agents path length is not linear bounded to the obstacle perimeter length for Angulus and the agent can be forced to circle the signal source an arbitrary number of times only by adjusting the shape of the obstacle.

Figure 3.10 summarizes the differences between the Angulus algorithm and our solution. Note that we already pointed out in the previous section that there are types of obstacles for which both algorithms show the same, unbounded behavior. While the Angulus algorithm behaves also unbounded on spirals around the signal source, Algorithm 3.1 solves those problems with a path length bounded by obstacle perimeter and start/signal source distance. Environments with lots of leave points on the other hand force an agent to walk very long ways if our solution is implemented whereas Angulus will give more efficient results.

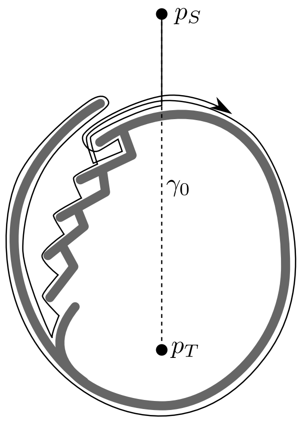
We conclude that the presented algorithm is usable as an alternative to the Angulus algorithm especially for environments in whose the signal source is enclosed by one or more spiral-like obstacles. Also - in contrast to Angulus - the set of potential obstacle leave points does not depend on the agents starting position but only on the environment. A notable weakness of our method though are environments with lots of leave points which can lead to long execution paths.



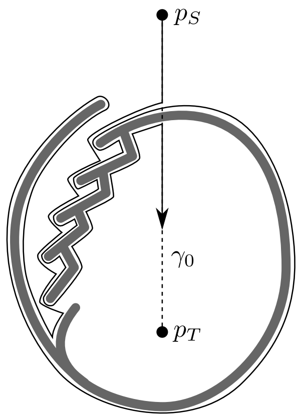
(a) Malicious case for Angulus



(b) Algorithm 3.1 in the same environment



(c) Malicious case for Algorithm 3.1



(d) The Angulus algorithm in the same environment

Figure 3.10: a) A malicious, source-circling environment that forces the Angulus algorithm to exhibit unbounded behavior.
b) The path length of algorithm 3.1 executed in the same environment is linear boundable by the sum of the obstacle perimeter length and the distance between p_S and p_T.
c) Algorithm 3.1 executing in a malicious source circling environment. The path length grows exponentially with the number of “steps” and thus can not be bounded by obstacle perimeter length and/or distance between start and signal source with a linear term. (Only a fraction of the complete path is shown here)
d) Execution of the Angulus algorithm in the malicious environment for Algorithm 3.1. The total path length is bounded by obstacle perimeter and the distance between starting position and signal source.

Chapter 4

Discrete signal intensities

In this chapter we will consider environments with discrete signal intensities. The chapter is divided into multiple sections. Section 4.1 describes the common basic problem model for such environments. The following sections each describe a certain variation of that general model and provide an algorithm which lets an agent find the signal source in an environment of the considered type.

4.1 Model

In the following sections we will use the notion of environments with discrete signal intensities.

4.1.1 Environment

As before we will understand an environment as a definition of the agents configuration space $\mathcal{C} = \mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{semi-free}} \cup \mathcal{C}_{\text{blocked}}$ and the position of the signal source p_T. Additionally we now demand an environment to define an intensity function $I : \mathcal{C} \to \{1, \ldots, n_I\}$ with $1 < n_I \in \mathbb{N}$. This function describes the intensity value $I(p)$ an agent measures at any point $p \in \mathcal{C}$ in the configuration space[1]. With such a discrete intensity function there might be large connected components in $\mathcal{C}$ that have identical intensity. As already noted in Section 2.3, in case the environment contains geometric obstacles, we will assume that there is only a finite amount of them and that they have a smooth closed curve of finite length as boundary each. Also as noted earlier we will assume that $\mathcal{C}_{\text{free}}$ is bounded by a smooth closed curve which - where not further specified - will be treated as an obstacle wall by the following algorithms.

Definition 4.1.1 (Intensity ring)**.** Given a configuration space $\mathcal{C}$ and an intensity function $I : \mathcal{C} \to \{1, \ldots, n_I\}$, $n_I \in \mathbb{N}$, the set of *intensity rings* in $\mathcal{C}$, $\mathcal{R}(I) \subseteq 2^{\mathcal{C}}$ is defined as:

$$\mathcal{R}(I) = \bigcup_{i \in \{1, \ldots, n_I\}} \Gamma\left(I^{-1}\left[\{i\}\right]\right)$$

[1]One could argue that it would be more appropriate to define intensities only for $\mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{semi-free}}$ instead of the complete configuration space $\mathcal{C}$, since the agent will not perceive intensities in $\mathcal{C}_{\text{blocked}}$ anyway. However with I being defined on $\mathcal{C}$, we will see the notion of intensity rings (see Definition 4.1.1) is much more natural.

Where $\Gamma(M)$ is the set of connected components in a set M and $I^{-1}[\{i\}]$ denotes the set of all points in $\mathcal{C}$ with intensity i. $\mathcal{R}(I)$ has the following properties:

- $\mathcal{R}(I)$ is a partition of $\mathcal{C}$
- For all elements $R \in \mathcal{R}(I)$ it is guaranteed that $\forall p_1, p_2 \in R : I(p_1) = I(p_2)$ holds, so we will write $I(R)$ for short when we referring to the intensity of any point in the intensity ring R.

In the rest of this document we will only consider environments with a finite number of intensity rings. Furthermore we will assume that the boundary ∂R exists for all $R \in \mathcal{R}(I)$ and is a smooth, closed curve.

4.1.2 Mobile agent

Due to the discrete nature of the signal and the resulting fields of equal intensity, the agent will be forced to use covering. The agent will (partially) cover the environment searching for increasing signal intensities and/or the signal source. Naturally a point shaped agent can not visit every point in a given two-dimensional area in finite time, which is why we alter the definition of the mobile agent to include a covering radius $r \in \mathbb{R}^+$. In our model the agent can perceive all signal intensities and the signal source in a circle of radius r around its current position. Note that the agent can still perceive obstacles only by touching them.

This is equivalent to having no covering radius at all but instead having an environment with a certain "granularity". If the signal is coarse enough, the agent does not actually need to be able to perceive all intensities in an r-radius around its position. However describing the model that way around would make the handling of the intensity distribution function utterly complex, so we stick with the covering radius r.

This model can also be interpreted as enforcing a certain granularity for signal intensity distribution having the agent still only perceive a single point at once which might be the more common case.

An agent moving in an environment with discrete signal intensities is assumed to have the following capabilities:

(1) At any point during execution, the agent can determine its position relative to an arbitrary but fix point using for instance odometry.

(2) The agent can turn arbitrary angles and walk arbitrary paths in free space.

(3) The agent has the ability to detect contact with an obstacle during motion and to follow an obstacle wall.

(4) At any point the agent is able to detect all intensity values and the signal source in a radius $r \in \mathbb{R}^+$ around itself. More precisely, if $p_A \in \mathcal{C}$ is the agents current position, the agent has knowledge about $I(p)$ for any point p in $\{p \in \mathcal{C} |\; \|p - p_A\| < r\}$. Additionally if the source is located in that r-circle ($\|p_T - p_A\| < r$ holds), the agent will know the position of the radio source relative to its own position.

4.2 Circular intensity environments

4.2.1 Model

We now consider the case where the signal intensity distribution depicts a circular pattern around the signal source. We extend the definition of the agent in Section 4.1 by means to track its own rotations in a Pledge-like way (compare Pledge introduction in Section 2.4) meaning it tracks its orientation $\alpha \in \mathbb{R}$ without calculating modulo 2π as it would be usual for angles in other situations. For instance if the agents orientation is $\alpha_1 = 2\pi - \varepsilon$ and the agent does a counter-clockwise turn about 2ε, its new orientation is $\alpha_2 = 2\pi + \varepsilon$ and not ε.

Also an agent in this model is able to communicate with other agents in the same environment over arbitrary distances.

Definition 4.2.1 (Circular intensity environment)**.** An environment is called circular if its intensity function $I : \mathcal{C} \to \{1, \dots, n_I\}$ has the form

$$I(p) = \begin{cases} 1 & \text{for } \sqrt{p^2 - p_T^2} \geq a_1 \\ 2 & \text{for } a_1 > \sqrt{p^2 - p_T^2} \geq a_2 \\ 3 & \text{for } a_2 > \sqrt{p^2 - p_T^2} \geq a_3 \\ \vdots & \vdots \\ n_I & \text{for } a_{n_I - 1} > \sqrt{p^2 - p_T^2} \end{cases}$$

with $a_1 > a_2 > \cdots > a_{n_I-1} \in \mathbb{R}^+$. In other words the signal intensity for a point $p \in \mathcal{C}$ only depends on the distance of p from the signal source in a way that lower valued signals have a greater distance from the source, so all intensity borders are perfect circles.

4.2.2 Signal form following motion planning

The algorithm we are going to present is based on the idea that by knowing the signal form we can calculate the exact position of the signal source. With that information, algorithms like Bug1, Bug2 or I-Bug can be used to efficiently move an agent to the calculated location [1, 17]. For the calculation of the signal source position we assemble multiple points on the intensity borders (which are circles with unknown radii) and form an equation system that will be solved for these circles common center. In order to assemble these points, Algorithm4.1 controls multiple agents instead of one and uses the Pledge algorithm to distribute them on intensity border points.

After the agents have spread, there is information about n_p agent positions $p_1, \dots, p_{n_p}$ which lie on n_c different circles with different (not yet known) radii $b_1, \dots, b_{n_c} \in \{a_1, \dots, a_{n_I-1}\}$. Note that due to their ability to detect signal intensities, all agents know wihch circle they are on. Given these positions, an equation system can be set up like this:

$$\begin{aligned} b_1^2 &= (p_{Tx} - p_{1x})^2 + (p_{Ty} - p_{1y})^2 \\ b_2^2 &= (p_{Tx} - p_{2x})^2 + (p_{Ty} - p_{2y})^2 \\ &\vdots \\ b_{n_c}^2 &= \left(p_{Tx} - p_{n_p x}\right)^2 + \left(p_{Ty} - p_{n_p y}\right)^2 \end{aligned}$$

Algorithm 4.1 Signal form following motion planning (SiFF-MoP)

Preconditions:

- All agents start inside a ring with a not-minimal intensity (i.e. there is at least one additional intensity ring surrounding them)
- The algorithm is executed by enough agents so the position of the signal source can be calculated

```
execute Pledge algorithm until intensity border is reached
send position to other agents
wait for positions from other agents
calculate position of signal source
if closest agent to that point {
  execute bug1, bug2 or i-bug in order to reach point
}
else {
  aid moving agent in localization
}
```

The system consists of two unknowns (the position of the signal source p_{Tx} and p_{Ty}) plus one additional unknown for each circle considered (their radii b_1, b_2, ...). Each agent position in turn contributes an equation, so the position of the signal source is calculable if $n_p \geq n_c + 2$ holds.

As an example, consider the case where all agents start in the same intensity ring that has intensity $i \in \{1, \ldots, n_I\}$. This ring is enclosed by two intensity-border-circles with radii a_i and a_{i-1}. Since an agent stops as soon as it reaches such a circle in this case the agents will distribute on these two circles only, so $n_c \leq 2$ holds and the position of the source is calculable by four agents.

With the information of the sources exact position the agents can calculate their orientation and position in relation to the signal source at all times thus a wide range of known algorithms can be applied to find a way to the source.

Note that possible collisions between agents are not handled by the algorithm. This can be handled in different ways. For example, agents could execute the Pledge algorithm (as described in Section 2.4) in a fixed order so only one agent is moving at a time and can treat the others like normal obstacles. Another possibility is to implement more sophisticated "right of way" procedures that only force agents to wait when they are in acute risk to crash.

Theorem 4.2.2. *Consider a group of n_p agents executing Algorithm 4.1 in a circular intensity environment. One of these agents reaches the signal source after finite time if* $n_p \geq n_c + 2$ holds where n_c is the total number of intensity borders encountered.

Proof. We know that an agent executing the Pledge algorithm will eventually leave every finite area. Since there is at least one intensity ring surrounding each agent, all of them will eventually reach an intensity border. Because $n_p \geq n_c + 2$ holds, the equation system is non-singular and calculates the correct position of the signal source.

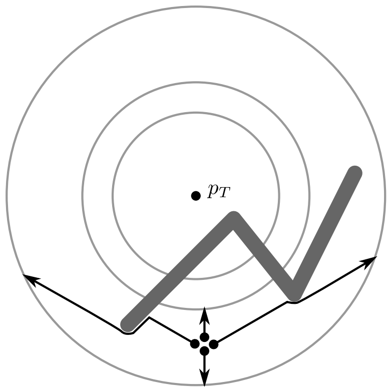


Figure 4.1: Sample run of Algorithm 4.1 (Distribution phase)

The algorithms “Bug1”, “Bug2” and “I-Bug” have been proven to lead their executing agents to the target in finite time too. □

4.3 Environments of arbitrary shape

In this section we extend the idea of covering using Morse decompositions as discussed in Section 2.6 so it accounts for varying signal intensities in the environment.

4.3.1 Model

In addition to the capabilities described in Section 4.1, we assume there is a Morse function $f : \mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{semi-free}} \to \mathbb{R}$ given so that $f(p) = 0$ if p is the starting position of the agent. Morse functions have been briefly discussed in Section 2.6. f defines the covering pattern for the agent and thus has to be chosen during implementation. Also in this model the agent is able to perceive tangents of obstacle walls and intensity rings in its covering radius if those tangents match that of f at its current position. For easier description of the algorithm we also assume that two curves $f^{-1}[y]$, $f^{-1}[y + c]$ with $y, c \in \mathbb{R}$ have a constant distance of c to each other.

4.3.2 Intensity aided coverage planning

In this model, the agent has so few information about the location of the signal source that it is unavoidable to have it cover potentially large parts of the free and semi-free configuration space. While covering the unknown environment, the free and semi-free configuration space is divided into cells, each of which is coverable with simple back-and-forth motions, much like it is the case for covering using Morse decompositions (briefly introduced in Section 2.6).

In contrast to a usual covering approach though, Algorithm 4.2 tries to minimize the area to be covered and focuses on areas with high signal intensity in order to locate the signal source quickly. We will first define the term cell in this context and then come up with the actual algorithm.

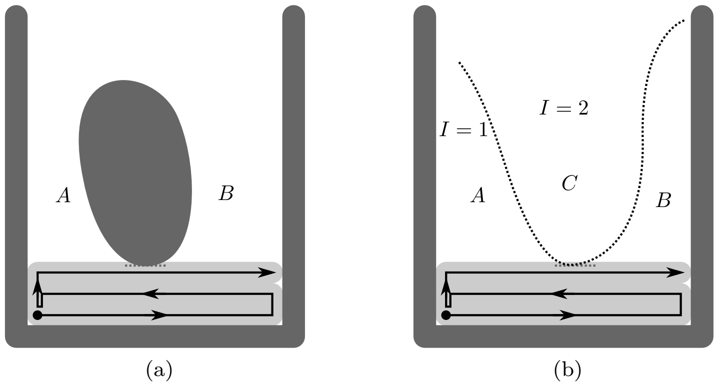


Figure 4.2: Intensity aided coverage planning compared to normal covering.
a) While covering, the agent encounters an obstacle tangent, closes the current cell and adds the newly found cells A and B to its list. The black line depicts the agents path, the light grey "background" under it the area being covered. The dark grey shapes represent obstacles, the dark dotted line implies the found wall tangent.
b) Similar situation with an area of different intensity instead of an obstacle. In contrast to the obstacle case, three new cells instead of two are found. In this example the new cell C has a higher intensity than A and B and will be covered before them.

Definition 4.3.1 (Simple cell)**.** A connected component $C \subseteq \mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{semi-free}}$ is called a *simple cell* with respect to a Morse function $f : \mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{semi-free}} \to \mathbb{R}$, iff $\forall a \in \mathbb{R} : (f^{-1}[a] \cap C)$ is connected.

Corollary 4.3.2. *Given a cell C which is simple w.r.t. a function f and a value $a \in \mathbb{R}$ so that $f^{-1}[a] \cap C \neq \emptyset$, the connected components of $C \backslash f^{-1}[a]$ are simple cells w.r.t. f.*

Algorithm 4.2 covers the environment with simple back-and-forth motions that we will describe in greater detail in Algorithm 4.4. The covering has completes a cell when it finds a signal intensity increase or an obstacle. In any case, newly found cells are added to $\mathcal{A}$ and a cell to be covered is selected from that set by the criterion of maximal signal intensity.

Algorithm 4.2 Intensity aided coverage planning (IACoP)

Preconditions:

- The agent is positioned at the outermost intensity border so that I(current position) = 1

```
k := 1
A_0 := {starting_area}
while |A_k| > 0 {
    C_k := argmax_{A ∈ A_{k-1}} I(A)
    N_k := cover(C_k)
    A_k := (A_{k-1} ∪ N_k) \ {C_k}
    k := k + 1
}
```

Given a Morse function f, and a cell C the function cover(C) (Algorithm 4.3) walks to the nearest known point in that cell, walks a complete back-and-forth-line (using function complete_line(), which will be described later on) and then starts to actually cover the cell using back-and-forth motions. If it found a wall- or intensity-ring-tangent it closes the current cell as shown in Figure 4.2.

Algorithm 4.3 cover() method for IACoP

Notes:

- f is the coverage-pattern defining Morse function chosen upfront
- p_A is always the momentary position of the agent

```
function cover($C$: cell) {
  $d$ := $R$
  walk to nearest point in $C$
  $y$ := $f(p_A)$

  // opening curve
  $\mathcal{N}$ := complete_line()
  if $\mathcal{N}$ contains area of higher intensity {
    return $\mathcal{N}$
  }

  // actual "back and forth" covering
  while $true$ {
    $y$ := $y + 2r$
    follow obstacle/intensity border
      until $f(p_A) = y$
    alternate $d$ between $R$ and $L$
    turn according to $d$ ($L$=left, $R$=right)
    walk curve so that $f(p_A) = y$ holds
    switch(what happened) {
      case found signal source {
        walk to signal source
        stop agent, terminate program
      }
      case found tangent {
        // closing curve
        $\mathcal{N}$ := $\mathcal{N} \cup$ complete_line()
        return $\mathcal{N}$
      }
    } // switch
  } // while
} // function
```

The complete_line() method is responsible for walking a complete line along constant value under f. In more simple words: Consider Figure 4.2 and imagine the agent does not start in the corner but in the middle bottom of the U-shaped obstacle. If it would just start the normal covering motion, there would be a piece left over on the left so it is necessary to walk a complete line which is exactly what that function does.

Algorithm 4.4 complete_line() method for IACoP

Notes:

- f is the coverage-pattern defining Morse function chosen upfront
- p_A is always the momentary position of the agent

```
function complete_line() {
  walk curve so that $f(p_A) = y$ holds
    until intensity change or wall encountered
  alternate $d$ between $R$ and $L$
  turn according to $d$ ($L$=left, $R$=right)
  walk curve so that $f(p_A) = y$ holds
    until intensity change or wall encountered
  return found areas that have not yet been covered
}
```

Observation 4.3.3. *An area C covered by Algorithm 4.3 is connected, hole-free, has a Jordan curve as boundary and is simple with respect to the used Morse function f. The boundary of C consists only of parts from obstacle boundaries, intensity ring boundaries and up to two curves in free space. We call such a curve* closing curve *of C if it is traveled after the covering of cell and* opening curve *else.*

Theorem 4.3.4. *Given an Environment with reachable radio source, Algorithm 4.2 reaches the radio source within a finite path length.*

Proof. Since number and circumference of obstacles and intensity rings are finite, there can also be only a finite number of critical points for any Morse function f on the environment. Thus the algorithm will split the environment into a finite number of cells, each of which will only take a finite time to cover (if it is covered at all). So every cell that is inserted into $\mathcal{A}$ will be covered at some point (except if the signal source is being found; in the meantime).

However because all reachable neighbor cells of each covered cell are added to $\mathcal{A}$ and the algorithm continuously covers cells in $\mathcal{A}$, it will eventually cover $\mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{semi-free}}$ completely if it does not encounter the signal source before. □

Theorem 4.3.5. *For any choice of $f : \mathcal{C}_{free} \cup \mathcal{C}_{semi\text{-}free}$ according to the modeling specification, any arbitrary (high) value $a \in \mathbb{R}^+$ and any arbitrary (low) value $\varepsilon \in \mathbb{R}^+$ there is an environment so that an agent executing Algorithm 4.2 will have covered an area of size a before it reaches the signal source. Additionally $a \geq |\mathcal{C}_{free} \cup \mathcal{C}_{semi\text{-}free}| - \varepsilon$ holds for that environment.*

Proof. We construct the environment in the following way: Depending on f we build an area of size a with the minimal intensity 1 so that the agent will cover a completely before reaching an area of higher intensity. We then "append" an area of intensity 2 and size ε to that area which contains the signal source. $\square$

We see from this last theorem that the general case suffers from the very loose demands we make to the environment which allow it to enforce arbitrary unwanted behavior on the agent. In Section 4.5 we will constrain the environment a little more so we can provide an upper bound for the agents path length.

4.4 Star shaped environments

4.4.1 Model

In addition to the properties described in Section 4.1 we demand from the environment that the signal distribution is *star shaped.* This kind of environment is characterized by the property that a ray from the radio source through any point of an intensity border does not subtend the same border at a second point, see Figure 4.3 for an illustration. Intuitively speaking this forbids "hooks" in intensity rings.

We also require an agent that is capable of determining tangents on intensity borders at its current position. In addition to the tangents themselves the agent must have the ability to detect which side of the tangent has points in the higher valued intensity ring (*inner side*). The agent must have the computing power to calculate and save the intersection of all detected inner sides.

Definition 4.4.1 (Star shaped environment)**.** Be p_T the position of the signal source in a nested environment and $\mathcal{R}(I)$ the set of the environments intensity rings. The environment is called star shaped iff

$$\forall R \in \mathcal{R}(I) : \forall p \in \partial R : \forall d \in \mathbb{R}^+ : (p_T + d(p - p_T) \in \partial R) \rightarrow (p_T + d(p - p_T) = p)$$

Note that this definition implies that star shaped intensity environments have exactly one hill.

4.4.2 Intensity aided coverage planning for star shaped environments

Due to their definition it is possible for each tangent on an intensity border in a star shaped environment to state the side of the tangent which the signal source must be located in order to not violate the constraint of Definition 4.4.1.

Given multiple tangents, we can use this information to define a polygon which must contain the signal source, called *kernel.* This approach even allows us under certain circumstances to skip covering parts of the maximal intensity ring (the mountain).

Definition 4.4.2 (Kernel of an intensity ring)**.** Let $R \in \mathcal{R}(I)$ an intensity ring and $\partial_o R \subseteq \partial R$ the smooth closed curve that represents the R's outer boundary[2].

[2]As a reminder: The complete boundary ∂R, can contain multilpe curves (in case R has "holes"), only the curve $\partial_O R$ encloses all points in R though.

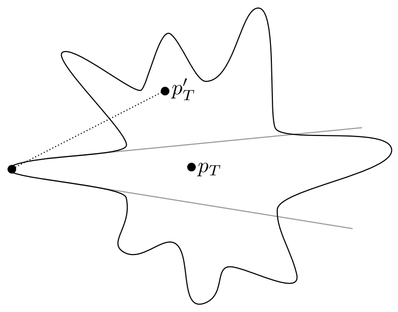


Figure 4.3: In a star shaped environment, intensity border tangents can be used to narrow down the location of the signal source. If the signal source would be on the other side of one of the tangents, for example at p'_T, the environment would not fulfill the star-shapedness constraint (dotted line).

A straight line g is called tangent of R in point $b \in \partial_o R$ iff it is a tangent of $\partial_o R$ in point b. Since g is a straight line in $\mathbb{R}^2$, it splits $\mathbb{R}^2$ into two half-planes from which exactly one contains points in R near b. We call the union of that half-plane and g the *inside* of g, the other half-plane is called *outside* of g.

Let T the set of all possible ring tangents in $\mathcal{R}(I)$ then for any $U \subseteq T$ we define the *kernel* $K_U \in \mathbb{R}^2$ as the intersection of the insides of all tangents in U. Further we define $K_\emptyset = \mathbb{R}^2$. Note that depending on U, K_U might or might not be bounded, since it is an intersection of half-planes however, it is always convex.

Theorem 4.4.3. *Given a star shaped environment $\mathcal{R}(I)$ with radio source at position $p_T \in \mathcal{C}$, be T the set of all possible ring tangents in $\mathcal{R}(I)$. Then $p_T \in K_U$ holds true for all $U \subseteq 2^T$.*

Proof. We prove by induction over $|U|$.

Basis For $|U| = 0$ is $K_U = \mathbb{R}^2$ by definition, thus $p_T \in K_U$ holds.

Step Let p_T be an element of K_U for some value of $|U|$. We define $U' = U \cup \{g\}$ with $g \in T \backslash U$. Assume, $p_T \notin K_{U'}$. Since $K_{U'} = K_U \backslash (\text{outside of } g)$ and $p_T \in K_U$ p_T must be in the outside of g. Let R be the ring of which g is a tangent and $b \in \partial_o R$ the touching point. p_T obviously can't be outside of R as the environment is nested, so it must be in R or an enclosed ring. Let $v : \mathbb{R}^+ \to \mathbb{R}^2$ with $v(d) = p_T + d(b - p_T)$ be a ray from p_T through $v(1) = b$. As p_T is in the outside of g, the ray approaches b from a ring enclosing R, however p_T is in R or an enclosed ring. Thus, there is a value $d_0 < 1$ so that $v(d) \in \partial_o R$ (see Figure 4.3). This contradicts Definition 4.4.1 $\lightning$. So $p_T \in K_U \to T \in K_{U \cup \{g\}}$holds for all $g \in T \backslash U$.

$\square$

Now we can modify the algorithm from the previous section to record all tangents found in a set U_k and to concentrate its search on the points in $K_{U_k} \cap (\mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{semi-free}})$. That allows the agent to cover a smaller area and still guarantees to eventually find the signal source.

Note that Algorithm 4.5 alters not only the “perceived” intensity value for specific cells during runtime but also changes the structure of the intensity ring

Algorithm 4.5 Intensity aided coverage planning for star shaped environments (IACoP-SE)

```
1 $k$ := 1
2 $\mathcal{A}_0$ := {starting_area}
3 while $|\mathcal{A}_k| > 0$ {
4   $C_k$ := $\underset{A \in \mathcal{A}_{k-1}}{\arg\max} I(A)$
5   $\mathcal{N}_k$ := cover $C_k$ tracking tangents
6   set $I(p)$:=$\lceil I(p) \rceil - \frac{1}{2}$ for all $p$ outside of kernel
7   $\mathcal{A}_k$ := $(\mathcal{A}_{k-1} \cup \mathcal{N}_k) \setminus \{C_k\}$
8   $k$ := $k$ + 1
9 }
```

graph, so it is no longer guaranteed that this graph is a tree at all times, even if it is when the algorithm starts.

Theorem 4.4.4. *An agent executing Algorithm 4.5 in a star shaped environment will reach the signal source in finite time.*

Proof. The proof is analogous to the general case (Theorem 4.3.4). Since the number and total boundary length of all obstacles and intensity rings is finite, so is the number of critical points the agent ever encounters. It follows that the environment will be split into a finite number of cells only, each of which can be covered in finite time. Since all neighbors of covered cells are added to $\mathcal{A}$ and all cells in $\mathcal{A}$ are covered eventually as long as the signal source is not encountered, the source is encountered after finite time if it is reachable.

□

Theorem 4.4.5. *For each value $n_I \in \mathbb{N}$ there is an environment so that Agorithm 4.5 covers $\mathcal{C}_{free} \cup \mathcal{C}_{semi\text{-}free}$ (almost) completely.*

Proof. We construct the environment as shown in Figure 4.4. Intensity ring borders in this kind of environment run either vertical or horizontal. In both cases tangents obtained from them do not change the structure of the intensity ring graph for $\mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{semi-free}}$ (the not yet covered part of $\mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{semi-free}}$ is completely inside the kernel at all times). Due to the layout of the environment as shown in the figure it is clear, that the agent will cover it completely before reaching p_T.

□

4.5 Environments of limited ring width

4.5.1 Model

In this section we will narrow down the specification given in Section 4.1. Additionally to the constraints given there we want to limit the width of intensity rings in the environment, see Definition 4.5.3 for a precise description. As in the general case presented in Section 4.3 the agent shall be able to perceive tangents of walls (only necessary for Section 4.5.4).

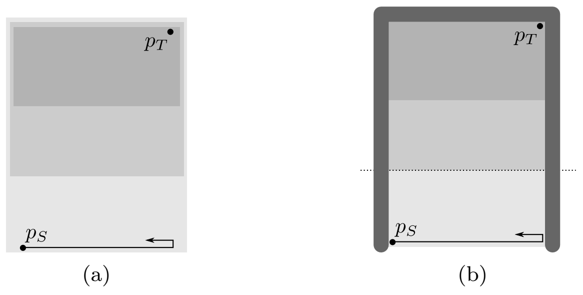


Figure 4.4: A malicious environment for Algorithm 4.5. a) Light gray tones indicate signal intensities. Note it is necessary for high-valued intensity rings to be a little more narrow than low-valued ones so they can be nested. b) An U-shaped obstacle “hides” the small passages on the side so the agent is forced to cover $\mathcal{C}_{\text{free}}$ completely. The dotted line is an example intensity ring tangent.

As we are considering hypothetical optimal online algorithms it is important to note that the agent knows about the ring width limit d (see below) but not about the structure of the environment, especially it does not know the number of intensity rings, n_I.

Definition 4.5.1 (Intensity adjacency graph)**.** Given a set of intensity rings $\mathcal{R}(I)$ (or just $\mathcal{R}$ for short if there are no ambiguities possible), we construct the intensity adjacency graph $G(\mathcal{R}) = (V_I, E_I)$ with
$V_I = \mathcal{R}$ and
$E_I = \{(a, b) \in \mathcal{R} \times \mathcal{R} \mid a \text{is adjacent to } b\}$.

Definition 4.5.2 (Nested environments)**.** An environment is called nested iff its intensity adjacency graph $G(\mathcal{R})$ is a tree and there is a node $S_{\mathcal{R}}$ so that for $G(\mathcal{R})$ rooted at $S_{\mathcal{R}}$, $\forall R_1, R_2 \in \mathcal{R} : (R_1 \text{ is a child node of } R_2) \Leftrightarrow (I(R_1) > I(R_2))$. Note that for each intensity adjacency graph there is at most one $S_{\mathcal{R}}$ that fulfills that condition and if it exists it is the node with the lowest intensity value. A leaf of a nested environments tree $G(\mathcal{R})$ is called hill of the environment.

Definition 4.5.3 (Ring width limit)**.** A nested environment with intensity rings $\mathcal{R}(I)$ has a ring width limit $d \geq 2r$ iff for all $R \in \mathcal{R}$ that are not hills

$$\forall p_l \in R : \exists p_h \in \mathcal{C} : (I(p_h) > I(p_l)) \wedge \left(|p_l - p_h|^2 \leq d\right)$$

holds and the hill which contains the source is contained in a circle around the source of radius d.

4.5.2 Lower bound for the case without geometric obstacles

In this section we are going to give a lower bound for the task of finding a signal source in a limited discrete environment without geometric obstacles. As a first step we provide a lower bound for the task of covering any given area completely as this will be an unavoidable sub-task for finding the signal source.

Lemma 4.5.4. *Be $A \subseteq \mathbb{R}^2$ an area, then an agent with a coverage radius of r must travel at least*

$$l_{covmin}(|A|) = \begin{cases} \frac{|A| - \pi r^2}{2r} & \textit{for } |A| \geq \pi r^2 \\ 0 & \textit{else} \end{cases}$$

in order to cover A completely.

Proof. First consider $|A| < \pi r^2$. Obviously, 0 is a lower bound for all coverage path lengths, so the claim holds true in that case. Additionally it might be the case that A fits completely in the scanning circle of the agent so no movement is required, which shows that this bound is actually sharp.

Now consider an agent which moves along a straight line of length a. The agent covers a total area of $|A_{\text{tube}}| = 2ra + \pi r^2$. Just before it starts moving, the agent already covers a circle with an area of πr^2. During movement it covers $2r\varepsilon$ for each movement step of length $\varepsilon > 0$, so the agent covers a total area of $2ra$ while in motion. Since the sum of this covered areas is exactly $|A_{\text{tube}}|$, we conclude that there is no “wasting” behavior, i.e. the movement of the agent could not possibly be altered in a way so that a path shorter than a would cover A_{tube} completely. So $l_{\text{covmin}}\left(|A_{\text{tube}}|\right) = \frac{2ra + \pi r^2 - \pi r^2}{2r} = a$ is indeed the shortest path length for coverage of A_{tube}.

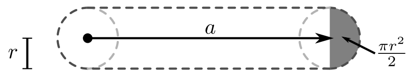


Figure 4.5: A_{tube}

As we already noticed in the example, the start position together with all movement steps sum up to exactly $|A_{\text{tube}}|$, so there is no “unnecessary” movement. That means there can be no area that could be covered more efficiently than A_{tube}. More precisely given any area A', an agent that covers A' travels $l_{\text{covmin}}(|A'|)$ or more depending on the shape of A'. We see that also in the case $|A| \geq \pi r^2$, $l_{\text{covmin}}(|A|)$ is indeed a lower bound for the coverage path length for A. Because A_{tube} can be covered with a path of length $l_{\text{covmin}}\left(|A_{\text{tube}}|\right)$ as shown above, also in this case the bound is sharp. □

Note that the optimal (offline) algorithm for the problem of finding the signal source has a maximum path length of $n_I d$. Since any online algorithm has to use covering at some point which involves a d^2-term, an online algorithm can not have a constant competitive factor. The next theorem gives a bound for online algorithms which is better suited for comparison.

Theorem 4.5.5. *For each online algorithm ONL that finds a radio source in ring width limited, nested environments with exactly one hill and no geometric obstacles, there exits such an environment $\mathcal{R}$ whose ring width is limited by $d \geq 2r$, with $n_I \geq 3 \in \mathbb{N}$ intensity rings so that an agent with a coverage radius of r executing ONL in $\mathcal{R}$ travels a total distance D_{ONL} bounded by:*

$$\begin{aligned} D_{ONL} &\geq \left(l_{1/2} + \pi(d-r)\right) n_I + \left(\frac{10\pi}{3} + 1\right) d + \frac{\pi}{2}\frac{d^2}{r} - \frac{23\pi}{6} r + l_{2/3} \\ &\geq \left(\left(\pi + \frac{1}{\sqrt{2}}\right) d - \left(\pi + \frac{3}{2}\right) r\right) n_I + \left(\frac{10\pi}{3} + 2\right) d + \frac{\pi}{2}\frac{d^2}{r} - \left(\frac{23\pi}{6} + \frac{3}{2}\right) r \end{aligned}$$

with $l_{2/3} = \sqrt{d^2 - 3dr + 3r^2} \geq d - \frac{3}{2}r$ *and*
$l_{1/2} = \frac{1}{\sqrt{2}}\sqrt{d^2 - \frac{r^3}{d} - 3dr + 5r^2} \geq \frac{1}{\sqrt{2}}\sqrt{d^2 + \frac{9}{4}r^2 - 3dr} = \frac{1}{\sqrt{2}}\left(d - \frac{3}{2}r\right).$

Proof. As a first step, the algorithm has to find the intensity ring R_2 with $I(R) = 2$. By definition of an environment with ring width limit d, this has to be in a maximum distance of d relative to the agents starting point. We place R_2 in a way that it overlaps the d-circle around the agents starting position only by a tiny bit at the point at the circles border that is covered last by the agent. This way the agent will always at least travel $d - r$ for getting to near enough to the border to scan it and at least $2\pi(d-r)$ in order to reach scan every other point on the circles border (see Figure 4.6a).

For R_3 the procedure is almost the same. However the last point on the d-circles border visited by the agent might be inside the previous d-circle. If we chose such a point for the "first" point of R_3, the agent could conclude that the signal source must be located inside the union of this two circles and just cover them and find the source early. Therefore, we do not force the agent to cover pieces of the previous circle again, Figure 4.6b illustrates the $\frac{2}{3}$-circle we are considering. Instead of the latest point on the whole circle we now place R_3 on the latest point on this $\frac{2}{3}$-circle, again forcing the agent to cover points on the border. This can not be done more efficiently for the agent than walking straight to the first point on the new $(d-r)$-circle (black line in Figure 4.7) and then walk the circle.

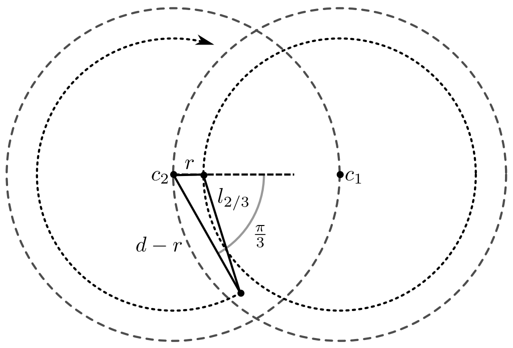


Figure 4.7: Transition to a $\frac{2}{3}$-circle

The length $l_{2/3}$ of that line is calculated using the cosine rule:

$$\begin{aligned} l_{2/3}^2 &= (d-r)^2 + r^2 - 2(d-r)r\cos\frac{\pi}{3} \\ &= d^2 + 3r^2 - 3dr \\ \Leftrightarrow l_{2/3} &= \sqrt{d^2 + 3r^2 - 3dr} \\ &\geq d - \frac{3}{2}r \end{aligned}$$

The length of the $\frac{2}{3}$ circle is given by $\frac{4\pi}{3}(d-r)$.

For R_k with $3 \leq k < n_I$ again we place a point of R_{k+1} so it is on the latest visited point on the d-circle border that is not inside of a previous d-circle. Note however that there might be two or more circles overlapping the current one. As Figure 4.6c shows, two of such circles can shorten the current circles

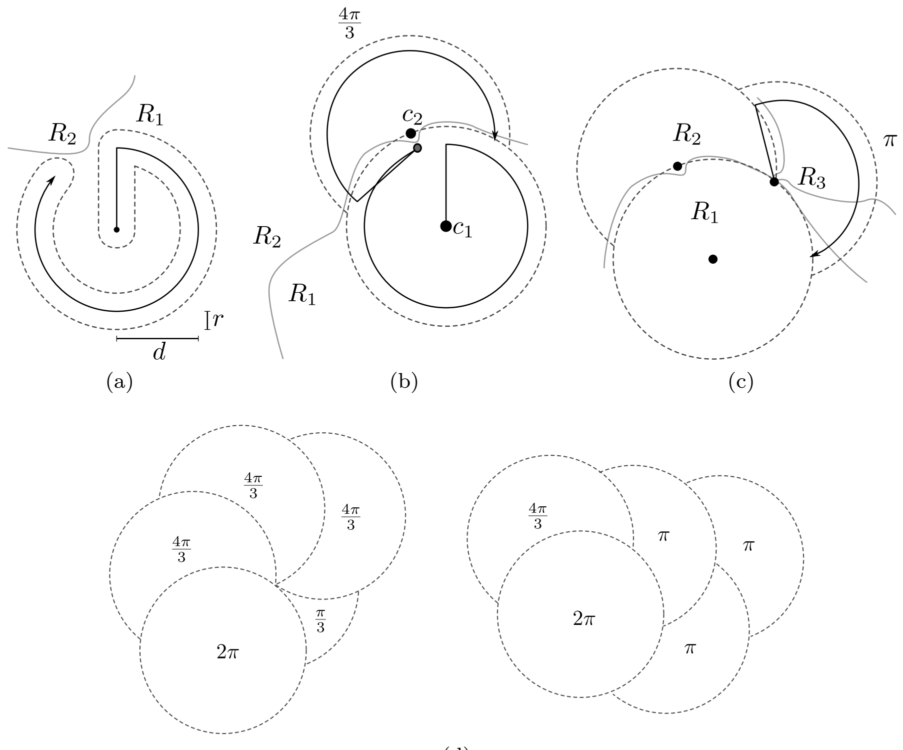


Figure 4.6: a) The intensity rings are placed in a way that force the agent to cover each point on the d-circle circumference in the first step.
b) In search for R_3 the agent needs to cover the circumference of a $\frac{2}{3}$-circle completely. If the "reachable" point of R_3 would be located on the inside of the first circle, that would play into the agents hands: The area is already partially covered and the agent might isolate that R_3 must be contained completely inside the circles, it could start covering earlier.
c) In following steps the agent can shorten the path length to a half-circle by visiting points near previous circles last.
d) Example illustration of the fact that its the best strategy for the agent to enforce half-circles. Even if shorter segments can be reached by choosing to walk $\frac{2}{3}$ circles before, half-circles are always cheaper in sum. (For the sake of a more comprehensible illustration we only depicted the d-circumferences here and left out the agents path and intensity borders)

"outer" circumference about half so the resulting length is π. If the agent would behave accordingly, there could also be cases where this distance goes down to $\frac{\pi}{3}$. However in order to achieve that the agent would have had to walk two $\frac{2}{3}$-circles instead of two $\frac{1}{2}$-circles before, so it would not shorten its path with such a technique (see Figure 4.6d). Besides in our model n_I is unknown to the agent so it could not know if its "investments" pay off.

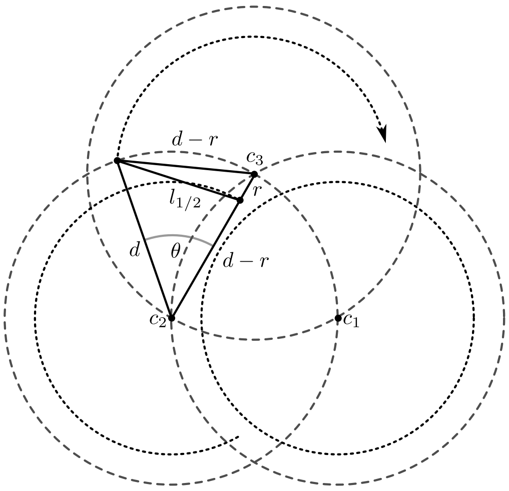


Figure 4.8: Transition to a $\frac{1}{2}$-circle

The transition cost for this case can again be calculated by using the cosine rule. Once for determining $\cos\theta$:

$$\begin{aligned}(d-r)^2 &= d^2+d^2-2d^2\cos\theta \\ \cos\theta &= \frac{-(d-r)^2+2d^2}{2d^2} \\ &= 1-\left(\frac{1}{2}-\frac{r}{2d}\right)^2\end{aligned}$$

And a second time for the actual calculation of $l_{1/2}$:

$$\begin{aligned}l_{1/2}^2 &= d^2+(d-r)^2-2d(d-r)\cos\theta \\ l_{1/2} &= \sqrt{d^2+(d-r)^2-2d(d-r)\left(1-\left(\frac{1}{2}-\frac{r}{2d}\right)^2\right)} \\ &= \frac{1}{\sqrt{2}}\sqrt{d^2-\frac{r^3}{d}-3dr+5r^2} \\ &\geq \frac{1}{\sqrt{2}}\left(d-\frac{3}{2}r\right)\end{aligned}$$

Eventually the agent will discover a point of the maximum intensity ring R_{n_I}. By definition the signal source must be contained in a d-circle around that point. We choose the last location in that area to be covered by the agent as position

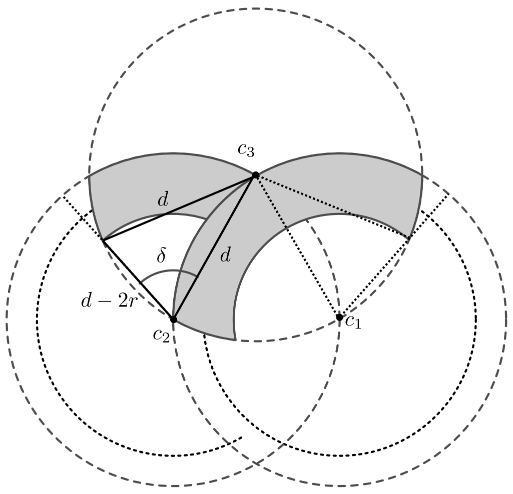


Figure 4.9: Calculating the upper bound for A_{done}

of the signal source, forcing the agent to cover the circle completely. Note that due to "forced" motions in the previous steps the agent has already covered

$$|A_{\text{done}}| \quad \leq \quad 3\frac{\delta}{2\pi}\left(\pi d^2 - \pi(d-2r)^2\right)$$

The cosine rule gives

$$\begin{aligned} d^2 &= d^2 + (d-2r)^2 - 2d(d-2r)\cos\delta \\ \Leftrightarrow \quad \delta &= \cos^{-1}\frac{(d-2r)^2}{2d(d-2r)} = \cos^{-1}\left(\frac{1}{2} - \frac{r}{d}\right) \end{aligned}$$

$$|A_{\text{done}}| \leq 12r\cos^{-1}\left(\frac{1}{2} - \frac{r}{d}\right)(d-r)$$

So the agent has to cover $\pi d^2 - |A_{\text{done}}|$ followed by r for actually contact with the source.

$$\begin{aligned} l_{n_I} \quad &\geq \quad l_{\text{covmin}}\left(\max\left\{\pi d^2 - |A_{\text{done}}|, 0\right\}\right) + r \\ &= \quad \max\left\{\frac{1}{2r}\left(\pi d^2 - 12r\cos^{-1}\left(\frac{1}{2} - \frac{r}{d}\right)(d-r) - \pi r^2\right) + r, r\right\} \\ &= \quad 6\cos^{-1}\left(\frac{1}{2} - \frac{r}{d}\right)(d-r) + \frac{\pi}{2}\frac{d^2}{r} + \left(-\frac{\pi}{2} + 1\right)r \\ &\leq \quad 3\pi d + \left(-\frac{7\pi}{2} + 1\right)r + \frac{\pi}{2}\frac{d^2}{r} \end{aligned}$$

In total the agent travels at least

$$\begin{aligned}
D_{\mathrm{ONL}} &\geq (2\pi+1)(d-r) + l_{2/3} + \frac{4\pi}{3}(d-r) \\
&\quad + (n_I - 3)\left(l_{1/2} + \pi(d-r)\right) + l_{n_I} \\
&= (2\pi+1)(d-r) + l_{2/3} + \frac{4\pi}{3}(d-r) \\
&\quad + (n_I - 3)\left(l_{1/2} + \pi(d-r)\right) + l_{n_I} \\
&= \left(l_{1/2} + \pi(d-r)\right) n_I + \left(\frac{10\pi}{3} + 1\right) d - \frac{23\pi}{6} r + \frac{\pi}{2}\frac{d^2}{r} + l_{2/3} \\
&\geq \left(\left(\pi + \frac{1}{\sqrt{2}}\right) d - \left(\pi + \frac{3}{2}\right) r\right) n_I \\
&\quad + \left(\frac{10\pi}{3} + 2\right) d + \frac{\pi}{2}\frac{d^2}{r} - \left(\frac{23\pi}{6} + \frac{3}{2}\right) r
\end{aligned}$$

□

4.5.3 Circular motion planning

In an environment of limited ring width $d \geq 2r$, we can be sure that there is a point with a higher intensity at most d away if we are not on a hill. Note however, this does not mean such a point is always reachable, as Definition 4.5.3 does not guarantee $p_h \in \mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{semi-free}}$. In this section we assume there are no geometric obstacles so $\mathcal{C} = \mathcal{C}_{\text{free}}$ holds. In order to take advantage of the ring width limit, we need a modified moving pattern so we can actually provide a bound for the agents travel distance depending on the ring width limit.

When an agent located at position c searches a ring of higher intensity we know that it must be located within a d-circle around c if the agent is not on a hill (in which case the signal source must be located in that circle). The idea behind our approach is: Without obstacles in the way, the agent can just try finding a ring of higher intensity by walking a d-circle around c. If that does not find a ring of higher intensity or the signal source, this will become a full circle. Since a potential ring of higher intensity must be in a d-radius around c, it must now be enclosed completely by the circle the agent just walked (this is because the intensity rings are connected areas). Because the environment is nested, any higher intensity ring and also the signal source must also be enclosed by that circle, so the source can now be found with simple covering.

See Algorithm 4.6 for a more technical description of this idea.

Theorem 4.5.6. *An agent executing Algorithm 4.6 will reach the radio source with a total maximum travel distance of*

$$D_{\mathrm{CMoP}} \leq ((2\pi+1)d - (2\pi+2)r)n_I + \frac{\pi}{2}\frac{d^2}{r} + (2\pi-1)d + (8\pi+4)r$$

Proof. When the agent starts executing the algorithm, it travels a straight line of length $d-r$, followed by at most a complete circle around c_0 of length $2\pi(d-r)$.

For the following circles around a point c_k, a point p of intensity $I(p) > I(c_k)$ can either be found during circling or not. If it is, a new circle is started around $c_{k+1} = p$ and the way costs for the circle around c_k are $d - 2r$ for getting to the closest point on the circle and not more than $2\pi(d-r)$ for actually walking it.

Algorithm 4.6 Circular motion planning (CMoP)

Preconditions:

- $n_I \geq 2$

```
$c_0$ := current position
$i$ := 0
walk straight length $d - r$
do {
  walk circle around $c_i$
  on intensity increase at $p$ {
    $i$ := $i + 1$
    $c_i$ := $p$
    walk to nearest point $q$ with $\|c_i - q\| = d - r$
  }
} until circle complete without intensity increase
drive covering spiral towards $c_i$
```

If the agent has not found a point with higher signal intensity than $I(c_k)$ this way, such a point must still be inside the d-circle around c_k by definition of a d-limited environment. Since the environment has only one hill, each of the inverse images of the possible intensity values under $I : \mathcal{C} \to \{1, \ldots, n_I\}$ is connected and thus in this case all points with intensity value higher than $I(c_k)$ can only be located within the d-circle around c_k which is especially true for the signal source.

The last step of the algorithm is always a (partial) shrinking spiral towards c_k followed by a motion of not more than r in order to contact the source.

$$
\begin{aligned}
D_{\text{CMoP}} &\leq (2\pi + 1)(d - r) + (n_I - 2)(d - 2r + 2\pi(d - r)) + \sum_{i=1}^{\left\lceil \frac{d}{2r} \right\rceil} 2\pi(2i + 1)r + r \\
&= (n_I - 2)((2\pi + 1)d - (2\pi + 2)r) + 4\pi r \sum_{i=1}^{\left\lceil \frac{d}{2r} \right\rceil} i + 2\pi r \left\lceil \frac{d}{2r} \right\rceil + (2\pi + 1)(d - r) + r \\
&= (n_I - 2)((2\pi + 1)d - (2\pi + 2)r) + 2\pi r \left\lceil \frac{d}{2r} \right\rceil \left(\left\lceil \frac{d}{2r} \right\rceil + 2 \right) + (2\pi + 1)d - 2\pi r \\
&\leq (n_I - 2)((2\pi + 1)d - (2\pi + 2)r) + 2\pi r \left(\frac{d}{2r} + 1 \right) \left(\frac{d}{2r} + 3 \right) + (2\pi + 1)d - 2\pi r \\
&= (n_I - 2)((2\pi + 1)d - (2\pi + 2)r) + \frac{\pi d^2}{2r} + 3\pi d + \pi d + 6\pi r + (2\pi + 1)d - 2\pi r \\
&= ((2\pi + 1)d - (2\pi + 2)r)n_I + \frac{\pi}{2} \frac{d^2}{r} + (2\pi - 1)d + (8\pi + 4)r
\end{aligned}
$$

□

4.5.4 Circular coverage planning

In the previous section we showed that in a d-limited environment without geometric obstacles it is always possible to find an area of higher signal intensity by walking a circle with radius d (d-circle for short).

When adding obstacles to the limited environment, we can not guarantee anymore that it is possible for the agent to walk that circle without being interrupted by a wall. We alter the procedure to have the agent walk spirals of increasing radius (up to d). If the agent is interrupted by an obstacle wall, it already has covered a circular area. Like in Section 4.3, the agent will split the environment at the wall and cover the resulting cells individually. It might be the case though, that a part of the d-circle around the agents starting position can only be reached by circling an obstacle. In order to take care of that, the agent will keep a list of encountered obstacle wall pieces. When it has finished covering all cells that were directly reachable, the agent picks an obstacle wall from its list and follows it until it reaches an uncovered part of the d-circle and continues covering. Since the environment is limited, after having covered the d-circle completely the agent will have discovered an area of higher signal intensity, where it starts a new circle.

Note that it is not possible to provide a single Morse function that describes this covering pattern completely as the circle centers are not known in advance but depend on the intensity distribution. That is why we introduce a more general form of function here.

Definition 4.5.7 (Rooted function)**.** Be $\mathcal{C}_{\text{root}} \subseteq (\mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{semi-free}})$ a set of rooting points and $g : \mathbb{R}^2 \to \mathbb{R}$ an arbitrary function. Then, for all $s \in \mathcal{C}_{\text{root}}$ we call a function $f_s : \mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{semi-free}} \to \mathbb{R}$ with $f_s(p) = g(p - s)$ rooted at s.

Corollary 4.5.8. *With g, f_s as in Definition 4.5.7, $f_s(p)$ is a Morse function for all $s \in \mathcal{C}_{root}$, $p \in \mathcal{C}_{free} \cup \mathcal{C}_{semi\text{-}free}$ iff $g(p - s)$ is a Morse function for all elements of $\{x \in \mathbb{R}^2 | \exists s \in \mathcal{C} : \exists p \in (\mathcal{C}_{free} \cup \mathcal{C}_{semi\text{-}free}) : x = p - s\}$.*

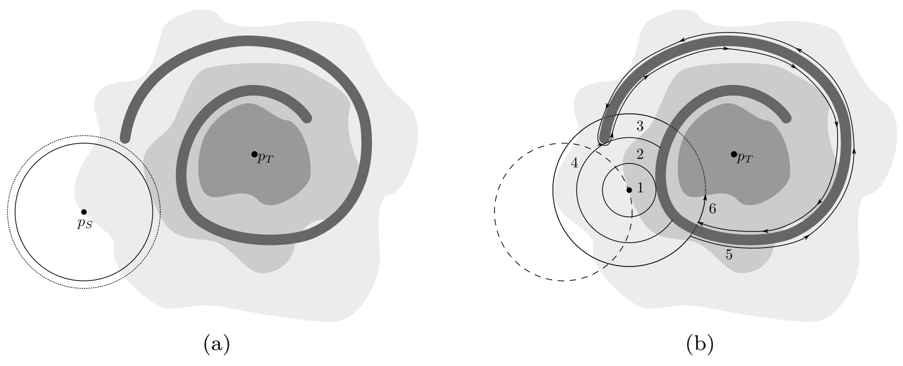
Algorithm 4.7 formalizes the discussed procedure.

Algorithm 4.7 Circular coverage planning (CCoP)

Notes:

- $f_c(p) = \|p - c\|$ is a Morse function if c is a point on an intensity border
- p_A is always the current position of the agent

```
$i$ := 1
$c_1$ := $p_A$
mark area in $d$-radius around $c_1$ as "to search"
$\mathcal{N}$ := {new cell at current position}
$\mathcal{W}$ := $\emptyset$
while $\exists N \in \mathcal{N}$ {
  cover $N$ with circles $f_{c_i}(p_A) = \{r, 3r, 5r, \dots\}$
  switch(what happened) {
    case found signal source {
      walk straight to signal source
      stop execution
    }
    case found point $p$ with higher intensity {
      clear "to search" markings
      $i$ := $i + 1$
      $c_i$ := $p$
      mark area in $d$-radius around $c_i$ as "to search"
      $\mathcal{N}$ := {new cell at current position}
    }
    case distance to $c_i$ is $> d$ {
      close current cell and remove from $\mathcal{N}$
    }
    case found wall tangent {
      add wall piece to $\mathcal{W}$
      do split/close walk
      remove closed cell from $\mathcal{N}$
      add found unknown cells to $\mathcal{N}$
    }
    case $\mathcal{N} = \emptyset$ {
      walk to next wall piece in $\mathcal{W}$
      turn right and follow wall
      thereby combine connected wall pieces into one
      when obstacle circled completely {
        mark circled area as covered
        remove wall piece form $\mathcal{W}$
      }
      when reached "to search" area {
        start new cell and add to $\mathcal{N}$
      }
    } // case
  } // switch
} // while
```



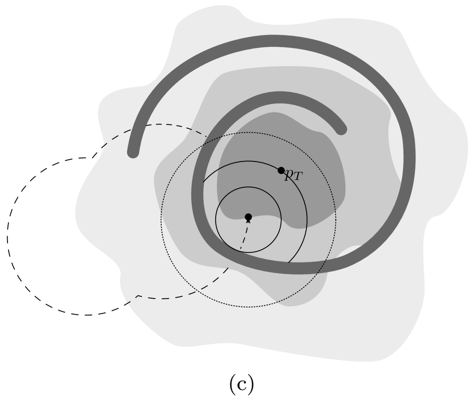


Figure 4.10: Example run of Algorithm 4.7.
a) The agent starts at p_S and covers the circle depicted by the solid line until it finds an area of higher intensity. The dotted line describes the d-circle around p_S.
b) The dashed line indicates the part already covered in (a).
b1) The agent covers a circle and splits the environment at the obstacle wall. The obstacle position is saved to a list for potential later circling.
b2) The agent covers the new cell again splitting the environment as it encounters an obstacle. To the agent there is no connection to the obstacle in b1 known, so it is added to the list as new obstacle.
b3,4) The agent covers cells 3 and 4 stopping when it reaches the d-circle. During covering the agent tracks obstacle walls so it concludes there are two obstacle pieces which touch multiple cells each (2,3,4 and 1,2,3,4).
b5) The agent has covered as much of the d-circle as was reachable without leaving it and thus now follows the nearest obstacle piece. During that process it concludes that the two found obstacle pieces belong to the same obstacle.
b6) The agent enters the d-circle again and starts resuming its coverage pattern (although in this case it starts with radius d and decreases that instead of increasing the radius like it did on the other side of the obstacle. During its first circle, the agent discovers an area of higher intensity.
c) The agent covers a circular area until touching of the obstacle enforces a split. While covering the new cell the signal source is found.

Theorem 4.5.9. *Consider an agent executing Algorithm 4.7 in an environment with one intensity hill, $n_O \in \mathbb{N}$ obstacles $O_1, \dots, O_{n_O}$ with boundary lengths $\partial O_1, \dots, \partial O_{n_O} \in \mathbb{R}^+$ and a radio source that is reachable by the agent. The agent will reach the signal source by traveling no more than*

$$D_{CCoP} \leq \left((2\pi + 1)d + \frac{\pi}{2}\frac{d^2}{r} + (2\pi + 2)r \right) n_I + 2\sum_{i=1}^{n_O} |\partial O_i|$$

Proof. The agent starts with a covering loop around its starting position. The actual covering happens in $\lceil \frac{d}{2r} \rceil$ circles with radii $r, 3r, 5r, \dots \left(2\lceil \frac{d}{2r} \rceil - 1\right) r$. Then the circular part of the covering loops is bound by $\sum_{i=1}^{\lceil \frac{d}{2r} \rceil} ((2i-1)2\pi r)$.

In order to move from one covering circle to the next one the agent needs to walk a distance of at least $\left(2\lceil \frac{d}{2r} \rceil - 1\right) r$ if there are no obstacles. Additionally the agent will travel n_I path segments of not more than r each for getting from one spiral to the next or to the source respectively.

If there are obstacles that limit the covering loop, these transitions will happen on the obstacles and thus be bound by $\sum_{i=1}^{n_O} |\partial O_i|$. The finding of an unexplored part of the covering circle also travels along obstacle borders so it is also bounded by $\sum_{i=1}^{n_O} |\partial O_i|$.

In total we get □

$$\begin{aligned}
D_{\text{CCoP}} &\leq \left(\sum_{i=1}^{\lceil \frac{d}{2r} \rceil} ((2i-1)2\pi r) + 2\left\lceil \frac{d}{2r} \right\rceil r \right) n_I + 2\sum_{i=1}^{n_O} |\partial O_i| \\
&= \left(4\pi r \sum_{i=1}^{\lceil \frac{d}{2r} \rceil} i - 2\pi r \left\lceil \frac{d}{2r} \right\rceil + 2\left\lceil \frac{d}{2r} \right\rceil r \right) n_I + 2\sum_{i=1}^{n_O} |\partial O_i| \\
&= \left(2\pi r \left\lceil \frac{d}{2r} \right\rceil^2 + 2\left\lceil \frac{d}{2r} \right\rceil r \right) n_I + 2\sum_{i=1}^{n_O} |\partial O_i| \\
&\leq \left(2\pi r \left(\frac{d}{2r} + 1 \right)^2 + 2\left(\frac{d}{2r} + 1 \right) r \right) n_I + 2\sum_{i=1}^{n_O} |\partial O_i| \\
&= \left((2\pi + 1)d + \frac{\pi}{2}\frac{d^2}{r} + (2\pi + 2)r \right) n_I + 2\sum_{i=1}^{n_O} |\partial O_i|
\end{aligned}$$

4.5.5 Circular coverage planning II

We want to show an apparent improvement to the algorithm defined in the previous section. We will present a simple modification to Algorithm 4.7 which intuitively behaves better in terms of total travel distance.

When evaluating Algorithm 4.7 it is noticeable it tends to cover certain areas twice: When a circular area has been covered and a point with higher intensity has been found, the next circular area is being covered around that point, leading to the overlapping path of those circles being covered two times. We want to explore that room for improvement by definition of Algorithm 4.8.

Algorithm 4.8 Circular coverage planning 2 (CCoP2)

Execute Algorithm 4.7 but instead of marking full d-circles around the current position for covering, subtract from these circles the circular areas that already have been covered.

The rest of this section will be dedicated to the algorithms runtime. Since we did not change its behavior with respect to obstacle walls, no change is to be expected concerting the portion of the path length that depends on the obstacle boundary length. Therefore, we will focus on environments without obstacles. See the following definition for how environments we want to examine look exactly.

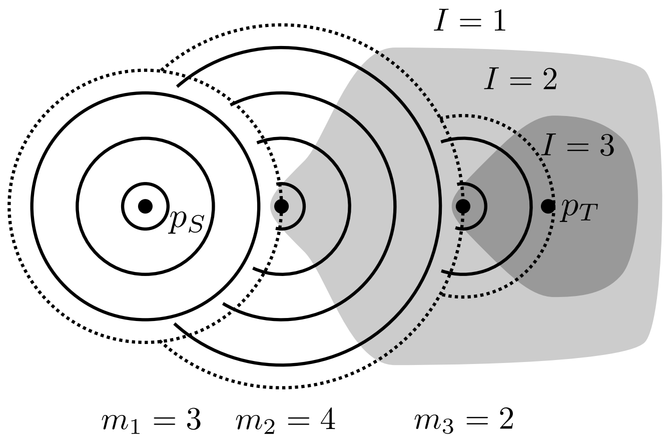


Figure 4.11: Example of an obstacleless environment for Algorithm 4.8. The (partial) circles represent covering loops of the agent (without loop transitions). The dotted line represents the border of the covered area. The gray tones indicate areas with different intensities.
The environment is constructed in a way that circular areas only overlap with their predecessor (i.e. it can not happen that 3 circular areas overlap), which makes calculations more manageable.

Definition 4.5.10 (Example environment for Algorithm 4.8.)**.** We define an environment without obstacles like shown in Figure 4.11. The concrete example in the figure can easily be extended to any number $n_I \in \mathbb{N}$ of intensity rings by being continued in the horizontal direction so that the center points of the covering circles all lay on a straight line.

The shape of the intensity rings will be chosen so that an agent executing Algorithm 4.8 has to walk $m_i \in \mathbb{N}$, $i \in \{1, \ldots, n_I\}$ covering loops to reach the $(i+1)$'th intensity ring (or the signal source if the agent is located on the hill). The choice of the parameters $m_1, \ldots, m_{n_I}$for this type of environment shall be constrained by the following conditions:

(1) A circular covering area is only allowed to intersect with its predecessor but not with its pre-predecessor, so only the previously covered circular area has to be subtracted when calculating the path length for an area.

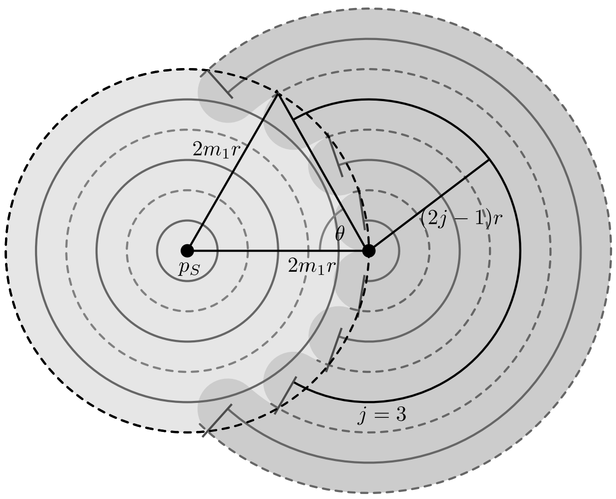


Figure 4.12: Geometry of overlapping circular covering areas

(2) No circular covering area is allowed to completely enclose another, i.e. $\forall i \in \{2, \ldots, n_I\} : m_i < 2m_{i-1}$ holds.

We can easily construct similar environments that force the agent to cover n_I circles with $m_1, m_2, \ldots, m_{n_I} \in \mathbb{N}$ covering loops.

Lemma 4.5.11. *The path an agent executing Algorithm 4.8 travels in an environment according to Definition 4.5.10 with enforced covering loop counts* $m_1, m_2, \ldots, m_{n_I} \in \mathbb{N}$ *has length*

$$\begin{aligned} D_{CCoP2}\left(m_1, \ldots, m_{n_I}\right) &= 2\pi r m_1^2 + 2rm_1 + \sum_{i=2}^{n_I} \Bigg(2\pi r m_i^2 + x_i \\ &\quad + \sum_{j=1}^{m_i} \left(\Delta(m_{i-1}, j) - (4j-2)r \cdot \cos^{-1} \frac{j}{2m_{i-1}} \right) + r \Bigg) \end{aligned}$$

With $\Delta(m_{i-1}, j) = r\sqrt{(8j^2+2) - (8j^2-2)\cos\left(\cos^{-1}\frac{j}{2m_{i-1}} - \cos^{-1}\frac{j+1}{2m_{i-1}}\right)}$ *and* $x_2, \ldots, x_{n_I} \in [0, 2r]$ *dependent on the environment, where* x_i *is the distance the agent travels from the last loop of the* $(i-1)$*'th circular covering area to the first loop of the i'th one.*

Proof. The total path length is the sum of path length for each circular area plus the distances $x_i \in [0, 2r]$ that are traveled when the agent found an area of higher intensity and walks to the starting point of the first loop (which has radius r) in the new circular area. Finally the agent needs to travel a distance of r to reach the found signal source.

$$D_{\mathrm{CCoP2}}\left(m_1, \ldots, m_{n_I}\right) = D^{\mathrm{full}}_{\mathrm{CCoP2}}(m_1) + \sum_{i=2}^{n_I} \left(x_i + D^{\mathrm{partial}}_{\mathrm{CCoP2}}(m_{i-1}, m_i) \right) + r$$

$D^{\mathrm{full}}_{\mathrm{CCoP2}}(m_1)$ consists of m_1 loops with radii r, $3r$, $5r$, … between these loops the agents needs to walk a distance of $2r$, which we will hence call loop transition.

$$\begin{aligned} D^{\text{full}}_{\text{CCoP2}}(m_1) &= \sum_{j=1}^{m_1} (2\pi(2j-1)r + 2r) \\ &= 2\pi r m_1^2 + 2rm_1 \end{aligned}$$

The following areas each have a piece "cut out" by the predecessor. Figure 4.12 shows this for the second area. The piece cut out for the j'th loop in the second area has length $(2j-1)\cdot r\cdot 2\theta(m_1, j)$. Where $\theta(j_1, j_2) = \cos^{-1}\frac{j_2}{2j_1}$. The transition distance for a circular area that has a piece cut out is a little more complex to describe. Figure 4.13 illustrates how the length of such a transition is calculated. $\Delta(m_{i-1}, j)$ is the distance the agent travels in the i'th circular area for transition from the j'th loop to the $(j+1)$'th loop.

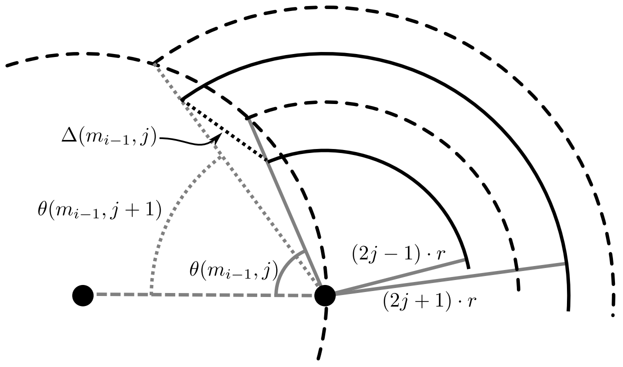


Figure 4.13: Calculation of $\Delta(m_1, j)$

$\Delta^2(m_{i-1}, j)$ is given by the cosine rule:

$$\begin{aligned} \Delta^2(m_{i-1}, j) &= ((2j-1)r)^2 + ((2j+1)r)^2 \\ &\quad -2r^2(2j-1)(2j+1)\cos\left(\theta(m_{i-1}, j) - \theta(m_{i-1}, j+1)\right) \\ &= r^2\left(8j^2+2\right) - r^2\left(8j^2-2\right)\cos\left(\cos^{-1}\frac{j}{2m_{i-1}} - \cos^{-1}\frac{j+1}{2m_{i-1}}\right) \\ \Delta(m_{i-1}, j) &= r\sqrt{(8j^2+2) - (8j^2-2)\cos\left(\cos^{-1}\frac{j}{2m_{i-1}} - \cos^{-1}\frac{j+1}{2m_{i-1}}\right)} \end{aligned}$$

We can also provide bounds for Δ:

$$\begin{aligned} \Delta(m_{i-1}, j) &\geq r\sqrt{(8j^2+2) - (8j^2-2)} \\ &= 2r \\ \Delta(m_{i-1}, j) &\leq r\sqrt{(8j^2+2) - (8j^2-2)\cos\left(\cos^{-1}\frac{1}{2} - \cos^{-1}\frac{2}{2}\right)} \\ &= r\sqrt{(8j^2+2) - (8j^2-2)\frac{1}{2}} \\ &= r\sqrt{4j^2-1} \\ &\leq 2rj \end{aligned}$$

Putting that together we get:

$$
\begin{aligned}
D_{\text{CCoP2}}\left(m_1,\ldots,m_{n_I}\right) &= 2\pi r m_1^2 + 2rm_1 + \sum_{i=2}^{n_I}\left(x_i + \sum_{j=1}^{m_i}(2\pi(2j-1)r \right.\\
&\qquad \left. -(2j-1)r\cdot 2\theta(m_{i-1},j) + \Delta(m_{i-1},j))\right) + r \\
&= 2\pi r m_1^2 + 2rm_1 \\
&\quad + \sum_{i=2}^{n_I}\left(x_i + 2\pi r m_i^2 + \sum_{j=1}^{m_i}\left(-(2j-1)r\cdot 2\cos^{-1}\frac{j}{2m_{i-1}} + \Delta(m_{i-1},j)\right)\right) + r \\
&= 2\pi r m_1^2 + 2rm_1 \\
&\quad + \sum_{i=2}^{n_I}\left(2\pi r m_i^2 + x_i + \sum_{j=1}^{m_i}\left(\Delta(m_{i-1},j) - (4j-2)r\cdot\cos^{-1}\frac{j}{2m_{i-1}}\right)\right) + r
\end{aligned}
$$

□

The interesting question arises, with given n_I and d, which combinations for values for $m_1,\ldots,m_{n_I}$ lead to the longest path. Recall from the previous section that $m_i \le \left\lceil \frac{d}{2r}\right\rceil$ must hold for any $i \in \{1,\ldots,n_I\}$ so that path can not be infinitely long. Intuitively, given a configuration $m_1,\ldots,m_x,\ldots,m_{n_I}$ with $m_x < \left\lceil \frac{d}{2r}\right\rceil$ one can create a more "expensive" configuration $m_1,\ldots,m_x+1,\ldots,m_{n_I}$, so $D_{\text{CCoP2}}(m_1,\ldots,m_x+1,\ldots,m_{n_I}) \ge D_{\text{CCoP2}}(m_1,\ldots,m_x+1,\ldots,m_{n_I})$ holds.

Informally speaking: Increasing the radius of one covering circle (where allowed) will increase the total path length so the configuration $m_1 = m_2 = \cdots = m_{n_I} = \left\lceil \frac{d}{2r}\right\rceil$ is the worst case for the presented algorithm. Although this hypothesis is supported by numerical path length calculations we were not able to come up with a lower bound for $D_{\text{CCoP2}}(m_1,\ldots,m_x+1,\ldots,m_{n_I}) - D_{\text{CCoP2}}(m_1,\ldots,m_x+1,\ldots,m_{n_I})$ that is guaranteed to be positive.

Claim 4.5.12. Be $D_{\text{CCoP2}}(m_1,\ldots,m_x+1,\ldots,m_{n_I})$ the path length of an execution of Algorithm 4.8 in an environment that conforms to Definition 4.5.10 with $m_x \in \left\{1,\ldots,\left\lceil \frac{d}{2r}\right\rceil - 1\right\}$ for an $x \in \{1,\ldots,n_I\}$ and $m_i \in \left\{1,\ldots,\left\lceil \frac{d}{2r}\right\rceil\right\}$ for $i \in \{1,\ldots,x-1,x+1,n_I\}$. Then

$$D_{\text{CCoP2}}(m_1,\ldots,m_x+1,\ldots,m_{n_I}) \ge D_{\text{CCoP2}}(m_1,\ldots,m_x,\ldots,m_{n_I})$$

holds.

Claim 4.5.13. In an environment conforming to Definition 4.5.10 with a ring width limit of d an agent executing Algorithm 4.8 will travel a path of length

$$
\begin{aligned}
D_{\text{CCoP2}}(m_1,\ldots,m_{n_I}) &\le \left(\left(\frac{4\pi}{3}+\frac{3}{2}\right)d + \left(\frac{\pi}{3}+\frac{1}{4}\right)\frac{d^2}{r} + \left(\frac{4\pi}{3}+4\right)r\right)n_I \\
&\quad + \left(\frac{\pi}{3}-\frac{1}{2}\right)d + \left(\frac{\pi}{12}-\frac{1}{4}\right)\frac{d^2}{r} + \left(\frac{\pi}{3}+1\right)r
\end{aligned}
$$

Note that direct comparison of this value with D_{CCoP} is not possible since not only both values are upper bounds but also D_{CCoP2} is a value for an environment without geometric obstacles. Nevertheless it is quite plausible that in an environment with obstacles the runtime of Algorithm 4.8 would be $D_{\text{CCoP2}} + 2\sum_{i=1}^{n_O} |\partial O_i|$, analog to D_{CCoP}.

It is open how a worst case environment for Algorithm 4.8 is shaped and thus, which upper bound for its path length can be given. We claim that such an environment enforces maximal covering circle radii i.e., $m_1 = m_2 = \cdots = m_{n_I} = \left\lceil \frac{d}{2r} \right\rceil$ which we showed turns out harder to prove than one might suspect. The lower bound given in the previous section will certainly hold for this variant, intuitively it is clear that Algorithm 4.8 is more efficient in terms of path length.

Chapter 5

Conclusion

We considered a variety of models for the problem of finding a signal source, which led to a number of approaches of different nature.

For the problem of finding a signal source using only a compass-like device and rotation tracking we have introduced a new approach that works different to the known Angulus algorithm in Chapter 3. Our method, though more complex in algorithmic terms, makes even lower demands on the agents abilities. In contrast to Angulus, the agent only needs to be able to leave an obstacle whenever it faces the source with our solution. The benefit of that being that all possible points are determined only by the shape of the obstacles but not by the agents starting position. Also with our approach an agent only executes turning motions that adjust its heading to an adjacent wall (assuming it faces the source when the algorithm starts). These two facts could make it possible to let the environment provide assistance for agents with imprecise hardware.

We have shown that like it is the case with Angulus, the time our algorithm needs to reach the signal source is not boundable by the sum of the obstacle perimeter lengths. The total path length for Angulus depends on the number of windings of a signal source circling obstacle. With our algorithm the path length depends on the number of leave points, thus it can serve as a time-saving alternative for spiral-like environments with relatively few leave points.

We could illustrate that known signal forms can allow exact calculation of the signal source position even when working with discrete intensities. This approach is very modular as it can be easily transferred to any signal form whose center is calculable from intensity border points. Also, due to factual calculation of the sources position, a huge number of motion planning algorithms can be used to actually guide the agent to the source. There is still room for ideas on how agents can efficiently spread out in order to find intensity borders without harmful interference of each others motions. Another open problem is a situation in which the agents are located on unlucky positions that lead to an unsolvable equation system. In this case a procedure would be useful that lets one of the agents find a different point on a known intensity border, which is especially challenging if the environment limits the agents motion at the border.

We introduced the general idea of our new combination of covering and motion planning that can be used to find signal sources with discrete signal intensities among geometric obstacles. We showed how known properties of the environment like star-shapedness of the intensity distribution could be incorporated into the algorithm in order to improve its runtime.

We gave a lower bound for online algorithms in limited environments without geometric obstacles and an example algorithm that does not quite reach that bound. However, we could demonstrate that for this setting an agent only needs to actually cover a circular area of fixed size and apart from that only travels a path of length linear to the number of possible intensities. It is not clear yet if the lower bound we presented is actually sharp, so further work could continue here.

For environments with geometric obstacles we provided a more covering-oriented approach. An improvement for that approach was provided and we conjectured an upper bound for its path length which depends on a claim about how a worst-case environment for this approach looks. It is however open if the proposed environment is indeed a worst-case and what the path length of an optimal online algorithm is.

Summarizing we could show that mobile agents can navigate in environments that provide very little information. We introduced a variety of different algorithms and with one exception delivered proofs about their runtime. There is some room for improvements regarding the bounded cases. Also a practical examination of our findings could provide further insight on the quality of the proposed algorithms.

Bibliography

[1] Harold Abelson and A. diSessa. *Turtle Geometry: The Computer as a Medium for Exploring Mathematics.* MIT Press, Cambridge, MA, 1981.

[2] Ercan U. Acar and Howie Choset. Sensor-based coverage of unknown environments. *I. J. Robotic Res.*, 21(4):345–366, 2002.

[3] Ercan U. Acar, Howie Choset, Alfred A. Rizzi, Prasad N. Atkar, and Douglas Hull. Morse decompositions for coverage tasks. *I. J. Robotic Res.*, 21(4):331–344, 2002.

[4] Dana Angluin, Jeffery Westbrook, and Wenhong Zhu. Robot navigation with distance queries. *SIAM Journal on Computing*, 30:2000, 2000.

[5] Glen E. Bredon. *Topology and geometry*, volume 139 of *Graduate Texts in Mathematics.* Springer-Verlag, New York, 1997. Corrected third printing of the 1993 original.

[6] M. Burger, Y. Landa, N. Tanushev, and R. Tsai. Discovering point sources in unknown environments. In *Proceedings of WAFR 2008*, 2008.

[7] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications.* Springer-Verlag, second edition, 2000.

[8] Ferreira and Ademar. Sensing, intelligence, motion: How robots and humans move in an unstructured world. *Pragmatics & Cognition*, 15(3), 2007.

[9] IEEE. Standard 802.15.4 for local and metropolitan area networks specific requirements part 15.4, 2006.

[10] Ishay Kamon and Ehud Rivlin. Sensory based motion planning with global proofs. In *IEEE/RSJ/GI International Conference on Intelligent Robots and Systems.*, volume 2, pages 435–440, 1995.

[11] Tom Kamphans and Elmar Langetepe. Leaving an unknown maze using an error-prone compass, 2006.

[12] Wolfgang Kühnel. *Differential Geometry: Curves - Surfaces - Manifolds.* American Mathematical Society, 2 edition, December 2005.

[13] Vladimir J. Lumelsky and Tim Skewis. A paradigm for incorporating vision in the robot navigation function. In *ICRA*, pages 734–739, 1988.

[14] Vladimir J. Lumelsky and Sanjay Tiwari. An algorithm for maze searching with azimuth input. In *ICRA*, pages 111–116, 1994.

[15] Gabor Pete. Morse theory, 1999.

[16] Lei Tang, Kuang-Ching Wang, Yong Huang, and Fangming Gu. Channel characterization and link quality assessment of ieee 802.15.4-compliant radio for factory environments. *IEEE Trans. Industrial Informatics*, 3(2):99–110, 2007.

[17] Kamilah Taylor and Steven M. LaValle. I-bug: An intensity-based bug algorithm. In *ICRA*, pages 3981–3986. IEEE, 2009.

Thanks Hella for all the help ♡