# SPIRAL SEARCH AS AN EFFICIENT MOBILE ROBOTIC SEARCH TECHNIQUE

SCOTT BURLINGTON AND GREGORY DUDEK
CENTER FOR INTELLIGENT MACHINES MCGILL UNIVERSITY
3480 RUE UNIVERSITY, MONTRÉAL (QC)
CANADA, H3A 2A7
{SCOTTYB, DUDEK}@CIM.MCGILL.CA

ABSTRACT. In this paper we consider the problem of robotic search: that is, having a mobile robot find a target object in an unknown environment with obstacles. As a side-effect, the robot explores the environment.

Based on previous results, we present a formal description of our problem and an algorithm to solve it. Our work is a derivative of spiral search techniques developed in more abstract contexts. We show that this algorithm has good worst-case performance, in terms of its competitive ratio. We also show experimental data validating the feasibility of our approach and typical results.

## 1. INTRODUCTION

Search is one of the fundamental tasks of life. Especially for artificial life. Numerous potential artificial intelligence problems have been cast as search problems and many of the most natural applications for mobile robots are search problems. Reasonable behaviour, even in natural settings, is the result of identifying and choosing an appropriate course of action from a very large set of possibilities. Humans and animals have uncanny abilities to make these decisions in an unpredictable world. In order for artificial agents to succeed, it is absolutely necessary to efficiently search through and quantify the decisions it is presented with.

In this paper, we consider the problem of automatically searching an unknown environment using a mobile robot. In contrast to classic AI search problems, this variant of search entails a penalty for moving between previously visited locations (what are traditionally described as nodes of a search tree). Prior work [1] has demonstrated theoretically optimal strategies and associated bounds on search in very simple environments. In addition, several authors have considered apatial search using heuristic rules or sensory ones, [4, 8, 7, 5, 6, 2].

Here, we elaborate on this theme and consider robotic search in complex worlds. In particular we provide an algorithm, analytical results in the form of a competitive ratio and simulation data illustrating how to perform robotic search in a planar environment with obstacles.

## 2. PRIOR RESULTS

2.1. **Exploration Strategies.** In this paper, we are interested primarily in searching a planar environment (or in practice a topological equivalent). The simplest deterministic methods may be those of breadth-first search [BFS] and depth-first search [DFS]. Somewhat more sophisticated methods depend on interative modifications: depth-first search [IDDF] and cost sensitive IDDF (known as

$A^*$ search). These methods, although popular and very general, each have restrictions that impair a mobile robot. Refer to *Algorithms* by [3] for a more detailed explanation than will be presented here.

Breadth-first search is often inappropriate to robotic search tasks as it incurs high overhead in terms of the amount of storage space required and too high an overhead in moving between previously visited locales. Physical motion being much more costly than computation for a mobile robot implies that a more judicious method will be favoured highly over one such as BFS that requires intensive motion. Consider for example searching a long (endless) hallway for a water fountain (assuming there is one in the hallway somewhere). A BFS searcher would spend all its time moving back and forth between opposite ends of the hallway.

Depth-first search avoids the shortcomings of BFS. It is very efficient in terms of the storage requirements of the algorithm. More importantly, in light of the difficulty of physical motion, nodes in the search tree are expanded according to their physical proximity during DFS, rather than their logical proximity as in BFS. This implication assures us of short physical motions for discovery of new territory. In bounded environments on $n$ vertices DFS requires at most $2n$ vertices for a complete traversal of the search tree. However in a potentially unbounded environment DFS fails badly in term of any definition of "reasonable behaviour".

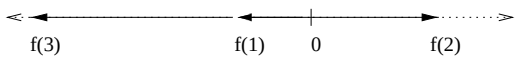
Iterated depth-first search conquers this shortcoming of DFS. IDDF maintains the advantages of DFS, while holding intact the *principle of locality* to some extent. There is a parameter which needs fixing here, when and how to we qualify the points of iteration. This is what typical $A^*$ search attempts to ascertain by assigning a cost-based iterating point. What can we use to generate such a cost in the complex world of a mobile robot? This is the insight that we offer. We suggest on the strength of planar search optimality in [1] that an efficient method of iteration may be dictated by *logarithmic spiral search.*

2.1.1. *Searching for a Point in a Line.* Back to our problem of finding the water fountain in an endless hallway. What is the best that we can hope for to guarantee that we find a drink regardless of which direction we choose to start down first?

We can describe the solution as a function $f(i) : Z \to Z$ where $f(i)$ reports the number of steps to take to the right or left of the origin before the $i$-th turn. This function, provided with the *progress* condition of,

$$f(i) \geq f(i-2) + 1 \tag{1}$$

is enough to specify a solution to the problem. The *progress* condition simply ensures that our solution will indeed explore new territory each time there is a change in direction. This is assured by taking at least one more step into unexplored territory than has been taken so that the search will terminate. Since this is an alternating search, $f(i)$ and $f(i+2)$ are consecutive distances from the origin on the same end of the line.




Baeza-Yates et al. [1] show that the following function,

$$f(i) = 2^i, \quad \forall i \geq 1 \tag{2}$$

results in an assymptotically optimal solution. With this function the total distance that is walked is $2\sum_{i=1}^{\lfloor \log n \rfloor + 1} 2^i + n$, since we walk out and back on every iteration of $2^i$ steps except for the last

| Info Given | Search Distance |
|---|---|
| Normal | $n$ |
| Distance, Slope | $3n$ |
| Distance, Horiz or Vert | $3\sqrt{2}n \approx 4.24n$ |
| Distance | $(1+\sqrt{3}+7\pi/6)n \approx 6.39n$ |
| Slope | $9n$ |
| Horiz or Vert | $13.02n$ |
| Nothing | $13.81n$ |

TABLE 1. Hierarchy of Search Distances bearing successive assumptions from [1]

probe, during which we walk $n$. This is easily seen to be bounded by $9n$ steps,

$$\begin{aligned} 2\sum_{i=1}^{\lfloor \log n \rfloor + 1} 2^i + n &< 2\left(2^{\lfloor \log n \rfloor + 2}\right) + n \\ &\leq 2\left(2^2 \cdot 2^{\lfloor \log n \rfloor}\right) + n \\ &= 2^3 n + n \\ &= 9n \end{aligned}$$

with a little bit of slack in the first inequality. In fact there exist algorithms which perform at $9n - \Theta(\log n)^i$, $\forall i$.

This particular problem, in a 2 dimensional setting is equivalent to looking for a line in a plane knowing only the slope of the line (we are along the normal for the intersection point).

2.1.2. *Searching the Plane.* It is worthwhile now to consider search techniques in the plane. The objective is to find some better method for searching in the plane. Bearing in mind that the environment is unknown (perhaps unbounded) in addition to the real problem of unreliable localization. Given the hierarchy collected by Beaza-Yates, Culberson and Rawlins in 2.1.2 we can get a flavour of how we may do better than DFS in a real environment.

Under the assumption that we have to pay a significant cost to "probe" in our environment proportional to the distance of the probe from our current location there may be better ways to search. Table 2.1.2 shows a collection of known best results for such search problems under the restrictions given.

2.1.3. *Logarithmic Spirals.* It is suggested by the results of 2.1.2 and by the rest of the findings in [1] that an appropriate way to proceed is through the continued application of these *logarithmic spirals*. We have chosen to use this approach to choose our iterating values for our IDDF.

There are a few things to mention about our representing a map by a tree structure. We make the representation of a polygonal environment by the search path that we would follow as seen from a perspective envisioning the environment as a whole. We presume that (in a hallway) the details of the search across the hallway, from side to side, are uninteresting as opposed to the progress of the search down the hallway. This perspective on the search progress allows a representation of the

polygonal environment as the set of search paths that are contained therein. Figure 1 shows how a star-shaped[1] polygon is represented as a set of concurrent (co-terminal) rays.

In an unbounded region our method still applies. The lack of a strict polygonal bound is not the issue here. In reality, with a limited range sensor, searching a local environment that is larger than the sensor range requires that we comb through the environment according to some tesselating pattern. The impact of the details of this part of the search will only turn up in the low order terms of the overall performance.

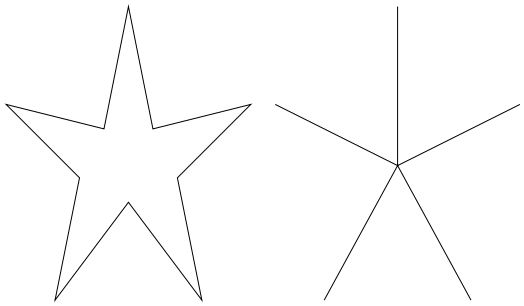


FIGURE 1. The representation of a star shaped polygon by a set of rays all co-terminal. This can be thought of as a potential search path.

Thence the definition of $m$-concurrent would be such a collection of $m$ rays. If we allow smooth deformations of this base search shape without allowing rays to cross over (creating cycles) then we can slacken this definition of concurrent rays to include homeomorphisms of concurrent rays.

Alas, all worlds are not this simple. It is possible that a path, and hence the associated ray, bifurcate. A discussion on which. The lifts the requirement that the rays be co-terminal. Even if the rays are allowed to bifurcate at points not at the origin, then the interpretation of spirally expanding search can still be applied.

A system capable of detecting cycles in the search tree would necessitates accurate odometry, which is beyond the scope of this simulation.

2.1.4. *M Concurrent Rays.* The case for searching $m$-concurrent rays is similar to that of searching for a point in a line, but offers further insight into that same problem. The assumptions are that the robot is begins at the intersection of $m$ rays and endeavours to find a target located on one of the rays. If the distance to the target is known, but not which ray the target lies within (comparable to knowing distance but not direction) then the obvious $(2m-1)n$ algorithm is optimal. If the distance is unknown, then this becomes the counterpart of the point-in-a-line problem and yields some interesting results.

It is in star-shaped environments [hulls] that this analysis can be directly adapted to provide both an algorithm and performance bounds. Given that there are $m$ rays, the order is not important since the names of any of the rays are simple labels and can be permuted, so cyclical indexing is fine from $\{m \in Z | m = 1, 2, \dots\}$. Later in the case of non-concurrent, or bifurcating, paths this issue come up

[1] A star-shaped polygon is one that contains a subregion known as a *kernel*, such that every point in the polygon is visible from any point in the kernel.

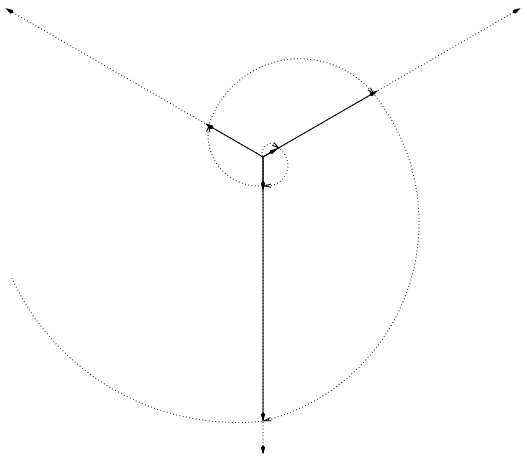


FIGURE 2. The path traced by $f(i) = \left(\frac{3}{2}\right)^i$ intersecting with $m = 3$ concurrent rays.

again, since the order of visitation will make a difference in the overall performance. Again we see that in order to study the problem, we have the *progress requirement* that the function, $f(i)$, which relates how far to walk from the origin before the $i-th$ turn is,

$$\begin{aligned} f(i) &= f(i-m)+1 \qquad \forall\, i \geq 1, \\ \text{where } f(-j) &= 0 \,\forall\, 0 \leq j \leq m-1 \end{aligned} \tag{3}$$

This has been termed by Baeza-Yates et al. as *generalized linear spiral search*, since the points we are interested in are the intersection points of the two sets in figure 2 and the actual spiral set which is the set of dotted points forming the logarithmic spiral depicted. The generalised linear spiral search algorithm is defined in a similar manner the linear spiral search from equation 2 as the number of steps to walk before the $i$th turn starting from the origin. Using the exploration function assures,

$$f(i) = \left(\frac{m}{m-1}\right)^i \qquad \forall\, i \geq 1. \tag{4}$$

optimal performance as shown in [1]. Thus we are instructed about what base to choose for the denominator in the underlying function. In the point-in-a-line case ($m = 2$) we have $f(i) = 2^i$, for the case of $m = 5$, for instance, we see that the function we should use is $f(i) = (5/4)^i$. This comes naturally from the proof (see [1]), with a worst performance ratio of,

$$1 + 2\frac{m^m}{(m-1)^{m-1}} \qquad \text{(for large } m\text{)}. \tag{5}$$

This is a *competitive ratio*. This is generated by determining the worst that this algorithm can possibly do in an environment chosen by an unfriendly arbitrator versus the best possible solution in that same environment.

## 3. OPTIMAL SEARCHING

3.1. **Searching M-Non-Concurrent Rays.** Demanding that we have concurrent rays is too much of a constraint to place on any system that we expect to use in the real world. Indeed we would like

to be able to use the techniques that the theory provides for us, but it must be adapted to actual use in many ways.

Concurrency of the search rays facilitates the analysis of the algorithm but is infeasible in practice. Typically we wish to use our robot to perform well in more complex environments. In particular the pathways that the robot is traversing may bifurcate. Figure 3 gives some indication how non-concurrent rays may model this type of world.

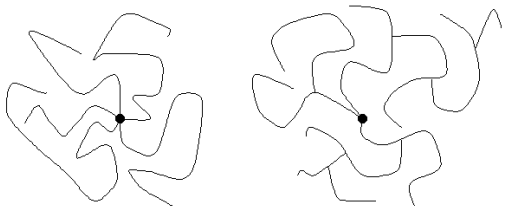


FIGURE 3. Examples of environments depicting concurrent and bifurcating paths.

The way that we propose to deal with this issue is to keep track of the base of each ray. That is, keep track of the distance travelled from the origin of each bifurcation in the ray structure. Since we have an exponential search algorithm, $f(i) = m^i$, the distance to the base of the ray is fixed and linear, and hence will be a low order term in the asymptotic performance of the algorithm.

## 4. THE EXPERIMENTAL APPROACH

4.1. **The Exploration.** In this section we present an experimental verification of the performance of our algorithm. This not only validates the worst case analysis, but indicates the types of results that can be expected in practice.

In the simulation presented here, we posit the existence of an idealised range sensor *with limited range*. The simulator returns accurate readings radially for a distance roughly equivalent to eight times the radius of the virtual agent. Our results are only weakly dependent on the faithfulness of the sensor in order to have an aspect of scalability to an actual robot.

The environment will grow in expanding concentric circles centered at the original location of the robot. The radial distances will reflect the current "search depth" that the robot is operating at. The concept that will be embedded here is the "Searching M-Concurrent Rays" idea, but with much more functional flexibility. Thus the robot wakes and begins looking about along these level curves of search depth. Initially the number of rays is 0. Figure 4 indicates the scenario thus far.

As the exploration begins the robot will work outwards along level curves. When an obstacle is encountered the path bifurcates to accommodate this and the number of concurrent rays is then incremented dynamically. Exploration continues as normal, only under the new circumstance of an increased number of rays. This impacts the distance down the current ray that we will search according to the theoretical optimal search ratio of $f(i) = \left(\frac{m}{m-1}\right)^i$.

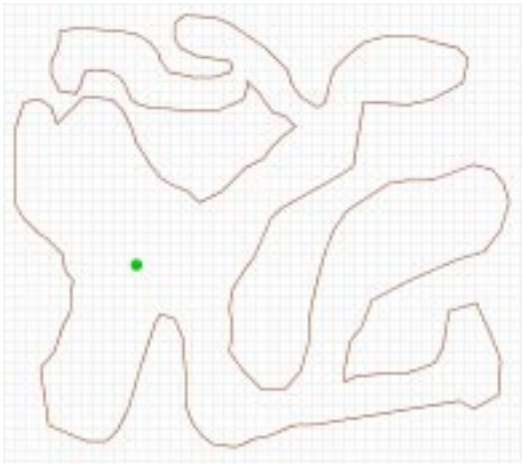


FIGURE 4. A possible scenario. The robot upon waking, however, knows nothing at all of the environment in which it finds itself.

As we progress through the environment, we see where the increments take place in figure 5. These are labelled in the diagram for clarity albeit incorrectly. They are incorrect in reality since *decrements* also occur when rays of the search have been exhausted[2].

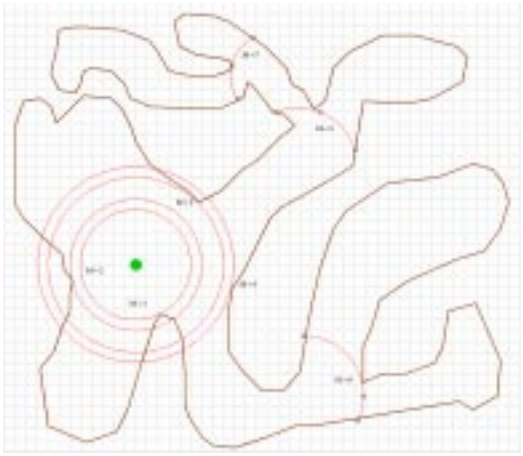


FIGURE 5. The concentric regions indicate the explored area at the time when the number of concurrent rays in increased. This occurs when an obstacle is encountered at a point in the middle of the exploration frontier within an area that was presumed to be a single ray, *ie* at a ray bifurcation.

[2]For instance the lower left region would certainly have been completely explored before the $M = 5$ vertex would be discovered and hence would have resulted in removal from the list of open rays to explore along with a decrement in the number of rays. So that by the time the $M = 5$ vertex is discovered there will actually be at most 3 active search rays, however this is a pedantic issue.

Thus we have an algorithm for extending the frontier of our search and we know that it is optimal under the ideal circumstance of concurrent rays. We have a method to adapt it to the non-concurrent situation that we are sure to run into in the real world, as our search paths bifurcate and unify, in order to guide us in expanding our knowledge of the environment in a locality-centric manner.

## 5. SIMULATION

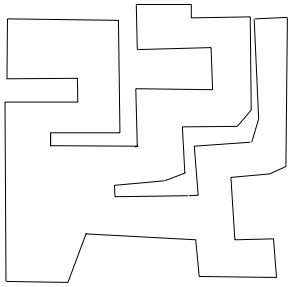
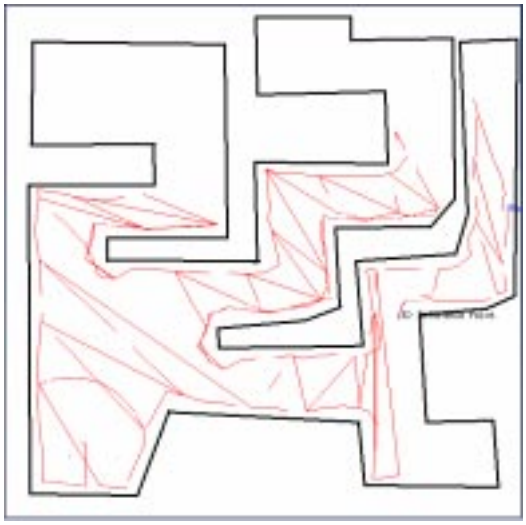


FIGURE 6. The original map which was used in the simulation.

Pictured are a series of iterations of a single run of our algorithm on a hand drawn, preconstructed map representing perhaps a set of hallways in a building. The scope of the entire map is roughly $200\times$ that of a real robot, for perspective sake. The original map is shown in figure 6.

The rest of the screen shots indicate the progress of the simulated robot at the end of each successive iteration. The extent of the map is not so large that a second cycle results in complete traversal of the environment. This brief exhibit should suffice to indicate the nature of the progression of the algorithm in simulation.



## 6. CONCLUSION

In this paper we have considered efficiently searching with a moving robot in a planar environment containing obstacles. This is an extension of existing work that proves the asymptotic optimality of spiral search in more abstract environments. Our work shows how to search a open or polygonal environment in a manner that is efficient in comparison to the optimal distance that would be travelling if the location of the goal were known in advance. This is accomplished by transforming environments that are homeomorphic to star-shaped polygon into sets of concurrent paths, while transforming other environments into sets of non-concurrent paths. These paths are then explored with a search radius that increases as a logarithmic spiral. In the experimental work we present, the feasibility and typical performance is illustrated.

There are several open questions posed by this work, some of which we are currently investigating. When this approach is used in practice, the accrual of positional error (for example via

odometry) is a real problem. This suggests that the optimal search strategy may involve returning to previously-explored regions slightly more frequently that might be suggested by the analysis presented here.

A related issue is that most real range sensors have an accuracy that diminishes with distance, and hence can be described in probabilistic terms. In some cases it may be preferable to search more deeply in regions were visibility is obstructed in order to assure a uniform coverage of free space along different paths. This suggests that probabilistic search may have a slightly different form from that described here.

Finally, we are currently examining parallel implementations of search for use by multiple robots: this leads to issues to how to subdivide the problem and how and when to merge partial results.

## REFERENCES

1. Ricardo A. Baeza-Yates, Joseph C. Culberson, and Gregory J. E. Rawlins, *Searching in the plane*, Information and Computation **106** (1993), no. 2, 234–252.
2. Tucker Balch and Ronald C. Arkin, *Communication in reactive multiagent robotic systems*, Autonomous Robots **1** (1994), no. 1, 27–52.
3. Thomas Cormen, Charles Leiserson, and Ronald Rivest, *Introduction to algorithms*, McGraw Hill, 1990.
4. Amitava Datta and Christian Icking, *Competitive Searching in a Generalized Street*, Proceedings of the Tenth Annual Symposium on Computational Geometry (Stony Brook, NY), ACM Press, June 6–8 1994, pp. 175–182.
5. Schmuel Gal, *Search games*, Academic Press, New York, 1980.
6. J. R. Isbell, *An optimal search pattern*, Naval Research Logistics Quarterly **8** (1957), 357–360.
7. Christos H. Papadimitriou and Mihalis Yannakakis, *Shortest paths without a map*, Theoretical Computer Science **84** (1991), no. 1, 127–150.
8. T. C. Schelling, *The strategy of conflict*, Harvard University Press, Cambridge, 1960.