WORST-CASE & AVERAGE-CASE EFFICIENCY TRADE-OFFS FOR SEARCH PROBLEMS

by

Preeti Sharma

Bachelor of Science, Chaudhry Charan Singh University, 1991

Master of Science, Chaudhry Charan Singh University, 1993

Master of Education, OISE University of Toronto, 2013

A thesis

presented to Ryerson University

in partial fulfilment

of the requirements for the Degree of

Master of Science

in the program of

Applied Mathematics

Toronto, Ontario, Canada, 2019

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public.

Worst-case & Average-case Efficiency Trade-offs for Search Problems

Master of Science, 2019

Preeti Sharma

Applied Mathematics

Ryerson University

Evacuation problems fall under the vast area of search theory and operations research. Problems of evacuation of two robots on a unit disc have been studied for an efficient evacuation time. Work done so far has focused on improving the 'worst-case' evacuation time with deterministic algorithms. We study the 'average-case' evacuation time (randomized algorithms) while considering the efficiency trade-off between worst-case and average-case costs. Our other contribution is to analyze average-case and worst-case costs for the cowpath problem (another search problem) which helped us to set a parallel method for the evacuation problem.

Acknowledgements

Many thanks to my supervisor Dr. Konstantinos Georgiou for his expert guidance with idea generation, coding and editing.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF APPENDICES

Chapter 1

INTRODUCTION

This section will familiarize the reader with a basic problem in the field of search theory as well as some basic concepts that will be used in the latter part of this thesis.

Consider the following simple sounding problem with two players, a hider and a searcher. Somewhere on a long path there is a hidden treasure placed by the hider. She does not want the treasure to be found and if found she wants to make sure that it will take as long as possible to be located. On the other hand, the searcher knows that there is a treasure somewhere on the path and she wants to locate it as soon as possible. She does not know how far is the treasure located or in which direction. In order to keep track of the distance and the direction, we shall call the starting point for the searcher as the 'origin'. Searcher will follow some plan (algorithm) and locate the treasure. It is implicit that among many possible algorithms some might be feasible while others may not be feasible. A possible algorithm could be to start search in one direction and search a portion of each side by travelling repeatedly and alternatively in both directions until the treasure is found. For this problem, we assume that the treasure is kept at a predetermined location by the hider and the searcher is given a predetermined algorithm to follow. The searcher follows the

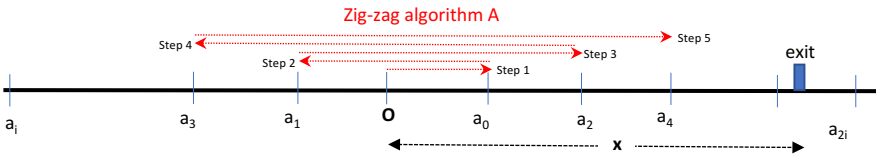

Figure 1.1: Basic Cowpath problem: a searcher to locate a treasure on a straight path.

zig-zag process until the treasure is located. We also assume that the searcher is moving at a constant speed (unit) and wastes no time when making turns. The treasure is located when the searcher and treasure are at the same location.

The problem, we just described, is a well-known problem in the field of computing and is referred to as Cowpath problem which was initially introduced in 1963 [17] and later reintroduced by computer scientists in the 1990s [60], [39]. While the searcher, the hider and the search space are the three main elements of the problem, there can be many variations to each of these elements. Therefore the extent of academic work in this simple sounding field is vast and very detailed, some of which will be referred to in the literature review section.

The searcher chooses an algorithm (search strategy) with the objective to find the treasure in the shortest possible time (compared to the worst-case time). In order to evaluate the performance of algorithms (with respect to the same input) search time appears to be a straightforward measure but it is not an effective measure, because the hider will put the treasure as far away as possible since she wants the treasure to be found as late as possible. For this reason we need to consider a *normalized* measure. If the searcher knows the location of the treasure, then she

can travel straight to the treasure via a shortest route. Time taken to complete this search is referred to as *offline performance* specific to that location of the treasure (input). When searcher does not know the location of the treasure and she follows an algorithm then time taken, to locate the treasure, is called *online performance* specific to the algorithm and the input. It is obvious that any algorithm will have its own worst-case online performance. The ratio of *online performance* and *offline performance* (for a specific input) indicates how well or poorly the algorithm has performed. Of all the possible ratios, the worst-case ratio is crucial in analyzing the performance of a plan and it is famously called the *Competitive Ratio (CR)*. The path followed by the searcher to locate the treasure to achieve the least-possible CR is referred to as the optimal search path. In a systematic process searcher will travel a certain distance across the origin and will gradually increase the step every time until the target is located.

The basic concepts of search theory are further expanded and are used intricately in the various search problems. Search problems have also taken the form of games over the years. It is now an umbrella term for the science of optimization and logical decision making in humans, and computers. Under the vast umbrella of optimization, search problems found utilities in many real-life scenarios. Some of the well known examples are in the military, search & rescue, scheduling, evacuation planning etc. For example military personnel can plan their combat strategy to optimally locate the enemy based on the terrain and availability of resources for a

higher rate of success in the combat & military missions. High security areas such as jails and official buildings can plan the routes, frequency and number of personnels for optimum surveillance so that the down time between each surveillance visit at any point in the entire area can be minimized. In the area of transportation, delivery truck-routes can be plotted and scheduled to improve delivery time and gas efficiency which can be used to assess the viability of the operations. Evacuation planning can be done in large areas where persons need to be evacuated in the situation of a disaster. Other examples may include robots trying to find a charging station; boats trying to locate a safe harbour in a storm, etc.

Many games were developed utilizing the search strategies as the player tries to minimize the possible loss for a worst case (maximum loss) scenario. Most games address a zero-sum game, in which one participant's gains results in losses for the other participant. If the total gains of the participants are added up and the total losses are subtracted, they will sum up to zero. These problems also paved the way for many games with variations on the basic problem such as network colouring game [24], high-low games [41], [57] , etc. Some famous problems related to optimal search strategies are Princess and Monster [50], Swimming in the fog [16], Submarine [65].

1.1 Thesis Organization

Chapter 1 continues to broadly explore the academic work done in the areas of 'search theory' and 'evacuation related problems' respectively. We look at some of the variations in the basic problems, that have been studied, and briefly touch on the related results. In this section, we also introduce some concepts that will be used in the latter part of the thesis. Chapter 2 specifically focuses on the Cowpath problem where we look at the trade-offs between the worst-case and average-case costs. It will lay out a structure of presenting the cowpath problem through a formal definition of the problem, worst-case cost, calculation of the average-case cost followed by the worst-case & average case analysis. Chapter 3 focuses on the specific evacuation problem to evacuate 2 robots from a search space (unit disc) that has the exit located on the perimeter using face-to-face communication. Similar to the structure in Chapter 2, Chapter 3 will present the problem, calculate average-case cost and discuss the results and their analysis.

1.2 Search Theory & Literature Review

While continuing to introduce more terminology used in the field of search theory, this section will familiarize the reader broadly with academic work done in this field.

We noted that the search problems have three elements, the Searcher, the Hider and the search space. The Searcher starts from origin O moving with a pre-determined speed and wishes to minimize the search time to locate the Hider by

following a search strategy. The intention of the Hider separates the basic problem into either a Search Game problem or a Rendezvous problem [8]. If the Hider does not want to be caught by the searcher, then she tries to maximize the search time. In Rendezvous problems, on the other hand, Hider wants to be found and as a result wants to minimize the search time.

The Searcher follows a search strategy and the Hider follows a hiding strategy. A strategy can either be deterministic (a predetermined algorithm) or randomized where player has the option to choose one strategy over another probabilistically. For randomized strategy, expected search time cost is calculated by taking probabilistic choices into consideration. In a feasible search strategy the Searcher moves back and forth across from the origin until the treasure is found with multiple turning points on both sides of the origin and a steady increase in each step, in order to cover all the points on the trajectory. The time spent to search the treasure is the cost and it is represented by a 'cost function'. If the Searcher starts by traveling distance a in first step then incremental increase in i^{th} step, given as a^i, will be optimal when the increase is exponential. Among many families of algorithms, the value of a is optimum at 2 for the optimal search trajectory. Beck and Newman [15] solved the linear search problem as a two-person zero-sum game. Gal [49] introduced the idea of a normalized cost function, also now referred to as the competitive ratio. A deterministic algorithm on Cow path problem guarantees a competitive ratio of 9 for capture time. This solution was obtained in the framework of an online algorithm

by Shmuel Gal [8]. Kao, Reif and Tate [60] used randomized search strategy and reduced the competitive ratio to 4.59. Baeza-Yates et. al [11] theorized a cost model for searching in any unbounded region that the cost is proportional to the distance of the object from the searcher's position. They also examined the effects of reducing the amount of information and determined that knowing the general direction of the object's location is much more informative than knowing the distance.

Alpern ([2], [3], [9], [5], [4]) and Gal ([6], [51], [52], [48], [13], [53], [54], [7], [69], [55]) contributed significantly to this area of search theory. The basic search problem has now been studied in great detail considering many scenarios with one or more variations in the three elements. Following are some of the examples of the variations that have been considered. The Hider can either be mobile or stationary. When the Hider is mobile then the hiding strategy can either be deterministic or randomized. Accordingly, the search algorithm changes. The Princess & Monster is a game with two participants, a monster who is a searcher looking to catch the princess, who is a mobile hider and does not want to be found by the monster. It was solved by Gal [50] and Lalley & Robbin [64]. Optimal hiding strategy is for the hider to move position not too often or rarely but this is not an effective strategy if hider knows searcher's position.

There have been many variations to the speed of the Searcher (unit speed in the original problem). In a search-and-rescue version of the problem, the searcher is supposed to bring hider back to the nearest exit in the network. Here the speed

of searching is not same as the return speed so, two-speeds [35] are taken into consideration. Other variations include considering turn-time where some time is added to the total cost when the robots make a turn [40]. In other scenarios, some of the Searcher robots are assumed to be faulty [31], [34] therefore they either provide no information or provide incorrect information. As a result the algorithm needs to confirm the information. These additional steps increase the time.

The search space variations include, a bounded or unbounded (original cow path problem) domain, a multi-dimensional region, a compact space or a network structure such as Eulerian network, tour (Chinese postman tour, Traveling Salesman tour etc.), one or more arcs (path between two points) [8], figure-8 network [8], a spectacle network [8] or unknown structure such as a maze [10], [8]. When considering search space as a graph with nodes and edges, some natural search strategies such as Chinese postman Tour, Random Chinese Postman tour, Traveling Salesman tour etc. have existed for a long time. Chinese postman Tour is a minimal length of a trajectory that covers all the edges and points in the search space and is a closed trajectory. Traveling Salesman Tour is a trajectory that visits all the nodes in the network and returns to origin in minimal time. Chinese Postman tour strategy used for an immobile hider is an optimal search strategy [53] in a weakly Eulerian network when the searcher can randomly move in any direction. Mathematicians and computer programmers have worked on many problems to find optimal search strategies in new networks.

Problems in the field of Patrolling/ Search, Cops & robber games and networking games have been explored in last few decades. 'Boundary / fence patrolling' problems have multiple agents protecting an area from invasion by an intruder who can enter the area from an unprotected point. They do not necessarily search for an intruder but the objective is to minimize the idle time (time between each visit), to reduce the uninspected time in the finite patrolling area leading to curb any intrusion. Czyzowicz et al [30] proposed the scenarios, with different number of agents, each with a maximal speed, as to which strategy is optimal. They also determined the scenarios where either of the two strategies does not provide the optimal idle time. While here each point is visited multiple times by the searcher, Beachcomber's problem [28] deals with scenario when a point is visited at least once and the focus is to have an algorithm for a fastest search considering two-speeds namely walking speed and searching speed. Other variations include the certain number of robots carrying out the search for a treasure or a hider; patrolling a fence [1], [25], [43], [42], [58], [66], [70] or an area; two-speed robots that have a walking speed and a patrolling speed, each of the multiple two-speed robots have speeds specific to them [30], [28], [35]; robots with distinct maximal speed [30]; two robots on a circular path with multiple exit points [27] search-and-fetch with two robots [56] etc.

Cops & robber type games were introduced in early 1980s where x cops are trying to capture a robber in a network that has nodes and edges. The smallest number of cops needed to catch a robber is called Cop number. The number of steps required

to capture make the length of the game. There have been many variations such as invisible robber [62], [23], partially invisible robber [18], use of various forms of detection such as traps, alarms, and photo radar [18], helicopter cops & robbers [46], game on a circular graph [45] , catching a fast robber on the grid [12] etc. Meyniel's conjecture postulates the upper bound of cop number $c(G)$ in a connected graph G with n- vertices to be $c(G) = \mathcal{O}(\sqrt{n})$. Frankl gave an asymptotic upper bound on the cop number [47]. Meyniel's conjecture has been proven to hold asymptotically for the binomial random graph [67]. Cop density of a finite graph, is defined as the ratio of the cop number and the number of vertices. In Cop & robber game played in infinite graphs, cop density is proven to be any real number in [0, 1] [19]. In a recent work in this area, Bonato et. al. [20] proved that the capture time decreases monotonically when more cops are added to the game play. Other variations include the game on a circular graph [45] , catching a fast robber on the grid [12] etc. While Fitzpatrick & Larken determined that on a circular graph, maximum cop number is 4, Balister et al determined the minimum number of cops required to catch the robber if robber's speed exceeds an absolute constant in an $n \times n$ grid [12].

Considering search spaces as graph with nodes and vertices, Network colouring and coloured coin games are another variation of search related problems. Introduced by Kearn et al [61], they model dynamic conflict resolution in social networks with the goal is to achieve a stable state in as less rounds of game as possible. In this model individuals are represented as nodes on the graph who only have local

information and they do not communicate with each other. Rossi & Ahmed [68] studied the large complex networks and developed a unified framework to be able to compare the methods across a wide range of social, web or biological networks. Greedy games are interesting variations to the search games where the hider can choose the amount of material to hide as opposed to the given quantity. Bigger amount is associated with the higher possibility of detection. High-low games [41], [57] are a variation of guessing game that have been studied. In Ed Gilbert's problem, hider is mobile between consecutive guesses and the secret number may change. Additionally the game is played in multiple rounds that remain the same.

In conclusion to the search, it can be said that we have come a long way from a simple problem in 1946 (Koopman [63]) to Network colouring problems (Kearn et al [61]) where complex networks are being studied to understand complex human behaviour. It is now a very popular area of inquiry contributing to the applications in the field of computer science, biology and economics.

1.3 Evacuation Problems & Literature Review

This section will familiarize the reader broadly with the academic work done in the field of evacuation type problems. Then we narrow down the focus on specific work done in the evacuation problem with 2 robots on a unit disc who communicate face-to-face.

Studied since 1990s by computer scientists, Evacuation problems are a specific

kind of search problems that have robots (or agents) searching for an exit to evacuate. Robots have information about the domain as well as the strategies of the other robots at the start. The goal is to minimize the time to reach the exit for all the robots in a worst-case scenario. Initial work considered evacuation planning as flow-problems [14], [59] on dynamic network with a starting point (source) and an end point (sink). The flow is governed by the distance and the time taken to cover the path between these two points. Various scenarios are considered for either the shortest time or the shortest path. Fekete et al. [44] explored the grid polygon search area with fixed exit as well as with mobile exit. They determined the worst-case bound between the fixed and the mobile exit scenarios. In the last decade, evacuation problems are studied with other variations such as multiple exit points [27], exit in a known or unknown domain [2], agents starting from a known [42], [29] or an unknown point, unit or different speeds of the searchers [35], [42], varied communication methods i.e. face-to-face (F2F) or wireless.

Communication among the agents is an important part of the strategy. It is assumed that Robots can not see each other. Currently two models are primarily studied in the field of evacuation, wireless model and face-to-face model (F2F). In face-to-face communication, they can only communicate the information to each other when they are at the same location. If they have devices that allow them to communicate with each other from a distance then the communication model is referred to as wireless communication.

'Search and fetch' variation of evacuation problems have searcher (one or more) trying to find the target in an unknown search space [2] and bring the target back with her. If there is more than one searcher then they communicate with each other either wirelessly or face-to-face. Georgiou et al. [56], [32] determined an upper bound in a wireless model as well as in a face-to-face model. For a face-to-face model they also determined a lower bound.

The variation of evacuation problems that we are interested in was introduced in 2014 by Czyzowicz et al. [29]. This problem had multiple (k) robots searching for an unknown exit point located at the perimeter of a unit circular disc. They found the lower bound for the optimal evacuation time in wireless as well as F2F communication methods and showed the difference in two methods for smaller values of k (2 & 3). For the larger values of k, they found almost-tight bounds on the asymptotic relation between evacuation time and team size. In a latter paper [27], Czyzowicz et al. studied the case with multiple (k) exits and two robots who are allowed to search freely. The exit is located on the perimeter of a unit disc. They tackled different scenarios i.e. two robots and 1 exit; 3 robots and 1 exit; k robots and multiple (4) exits on the perimeter. The communication model is wireless and robots are only allowed to move around the perimeter of the disc. They analyzed the algorithm where 2 robots start at the centre of the disc and then move to periphery together. Then they travel in the opposite directions to search for the exit. Once an exit is located by a robot then she travels to the other robot(s) to communicate

this information. Then they evacuate together from the exit. They determined the upper bound and lower bound for the cost when considering a single exit and k number of robots in a F2F model. For the two robots scenario the upper bound and the lower bound are calculated as ≈ 5.740 and ≈ 5.1999 respectively. They also determined both lower (≈ 4.826) and upper (≈ 4.826) bounds for the wireless model.

In the subsequent study Czyzowicz et al. [36] focused only on the evacuation of 2 robots from the unit disc using a F2F model. They introduced two new algorithms and were able to improve the upper bound to ≈ 5.628. They proved that if robots follow an algorithm with a detour even if the exit is not found then it improves the upper bound. First algorithm introduced a forced detour for the robots after a certain time when the exit is not found. The detour is a straightline path, inwards from the periphery. The robot returns back to the periphery via the same path and then continues the search on the periphery. Many families of algorithm are represented by $C(\chi, \phi)$ where χ is the distance travelled by the robot on the perimeter between 0 and π, and ϕ is the angle at which the detour starts. This algorithm improved the worst-case performance to ≈ 5.644 for specific values of χ and ϕ.

The second algorithm introduced a forced detour where the robot follows a triangular path. As a result, the algorithm has three points of significance in the optimal evacuation algorithm. Here also, the two robots are forced to meet after visiting a subset of vertices of the disc, even if an exit has not been found at that

time. Each family of algorithms is denoted by $C(\chi, \phi, \lambda)$ and λ is the time traveled by the robot in the disc. An optimum CR value ≈ 5.628 was achieved for the specific values of $\chi = 2.631865$, $\phi = 0.44916$ and $\lambda = 0.05762$. The lower bound was also improved to ≈ 5.255 from previous ≈ 5.199.

Brandt et al. [22] further improved the upper bound to ≈ 5.625 in the evacuation of 2 robots from the unit disc using a F2F model but the lower bound remained same at ≈ 5.255. This was done by forcing multiple detours for both the robots. This algorithm reduces the number of possible worst-case exits and this criterion can be applied to any area shape.

A "priority" version of evacuation [32], [33] was explored and solved in time at most ≈ 4.81854. There are 2 robots to be evacuated but one robot (queen) carries a higher priority for evacuation. The goal is to save a robot (queen) with higher priority and use their 'voice' to communicate. Both robots can be evacuated in ≈ 4.8264 by guiding the other person through their voice. The queen alone can exit in at most ≈ 4.81854. Any strategy for saving the priority-robot required at least ≈ 4.3896 in the worst case. For evacuating 2 robots, the time bound was improved to ≈ 3.8327 and ≈ 3.3738 for 3 robots. The lower bounds were calculated as ≈ 3.6307 and ≈ 3.2017 respectively for 2 and 3 robots. These results were significantly improved as compared to the previous work [29].

In the recent work, Czyzowicz et al. [37] bring together linear and circular search domains in the evacuation type problems for agents and provide a brief survey of

recent developments in group search and evacuation by a set of n co-operating, autonomous mobile agents. Both communication models between the mobile agents are being considered for the worst case analysis of the algorithms. Survey also covers the faulty agents, crash fault [38] and byzantine fault [34]. While a robot with byzantine fault communicates incorrect information, a robot with crash faults can stop movement and communication at any time.

While most evacuation type problems have focused on the worst-case scenarios, Chuangpishit et al. [26] started to explore the algorithms' average case analysis for the problem of evacuating two robots from the disc in the face-to-face model. They found that while worst-case performance of an algorithm improves, the average-case performance worsens for all the previously studied algorithms. They introduced new design of parameterized algorithm-families that minimize the average case cost of the evacuation algorithm while maintaining the worst case. They also made an observation that the average-case analysis is equivalent to designing randomized algorithms for this problem.

Chapter 2

COW PATH PROBLEM - WORST-CASE VS AVERAGE-CASE PERFORMANCE

This section initiates an average-case analysis for the previously described cow path problem and then compares the performance of the worst-case with the average-case.

To summarize the cow path problem, a stationary exit is placed at a point somewhere on a line by an adversary. Searcher S is unaware of the location of the exit and starts the search from origin O, moving at a maximal speed of 1. It can move in any direction on this unbounded search path. We assume that she can only locate the exit when she passes the exit and is at the same location as the exit. Searcher's objective is to locate the exit in the least possible time under worst-case scenario and she wastes no time when making turns. The zig-zag search algorithm (Figure 2.1) is the only feasible algorithm to locate the exit on a line path.

The Searcher announces the search strategy to the adversary, then adversary chooses the hiding strategy that will maximize the cost. The time spent in locating the exit is indicative of the cost, represented as $Cost(A, x)$, which is determined by the algorithm A (chosen by the Searcher) and x, the placement of the exit from the origin.

Here Searcher starts by travelling a_0 distance to the right in the 1^{st} step. If the

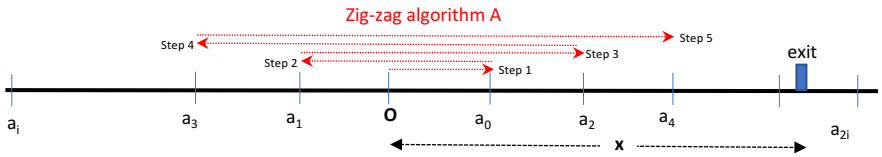

Figure 2.1: Zig-zag Path: Created due to repeated turning points in the algorithm. We further restrict the zig-zag algorithm A parametrized by an expansion factor α as A_α such that $a_i := \alpha^i$.

exit is not found then it travels back to the origin and continues to travel a_1 distance to the left of the origin. If the exit is not found then it travels back to the origin and continues to travel a_2 distance to the right of the origin. Searcher continues this multiple turn (zig-zag) algorithm in both the directions until the exit is found. We assume that the treasure is not kept very close to the origin otherwise small value of x will cause the competitive ratio to be arbitrarily high. When the Searcher starts from the origin then $i = 0$ then i increases every time the Searcher returns back to the origin. If i is even then Searcher travels a_i to the right else travels a_i to the left. It is imperative that the location of the turning point will be negative $(-a_i)$ for the odd value of i and positive (a_i) for the even value of i. Although the left and the right side are not identical but they behave symmetrically. Without loss of generality the analysis will be done on one side, as it can be shown that it is similar to the analysis for the other side so, for the ease of calculations and analysis, we assume that the exit is placed on the positive side of the axis.

It is clear that the Searcher will travel a_{2i} to the right and back to the origin or a_{2i+1} to the left and back to the origin, until the exit is found.

Observation 2.1. Certain requirements must be met for a zig-zag algorithm to be feasible:

- $a_i > 0$
- $a_i < a_{i+2} \ \forall \ i > 0$
- $a_{2i}, a_{2i+1} \to \infty$ as $i \to \infty$

We further restrict the zig-zag algorithm A parametrized by an expansion factor α as A_α such that $a_i := \alpha^i$. For any value of α, A_α will represent a specific algorithm. In this case, cost is calculated in terms of time spent to locate the exit.

Definition 2.2. *A_α is the family of algorithms that represent the zig-zag algorithm with α^i increment in the i^{th} step. $Cost(A_\alpha, x)$ is the cost incurred by the Searcher by following A_α algorithm where the exit is placed at distance x.*

2.1 Competitive analysis

Searcher's objective is to find the exit in the least possible time. Search space being an unbounded domain, the cost can be normalized as $\frac{Cost(A_\alpha,x)}{x}$. Determining the performance of the algorithm in the worst case scenario will help establish the benchmark for the least cost at the extreme end. *Competitive ratio* (CR) and *Competitive analysis* are standard notions [21] in the context of online algorithms in the area of computer science. We know that *CR* is the worst-case (supremum) ratio of the *online performance* and the *offline performance* and it indicates how well or poorly an algorithm performs as compared to the offline performance. For worst-case analysis, we find an algorithm A_α that will minimize the cost for the worst-case

scenario.

$$Wrs(A_\alpha) = \sup_{x>1}\left(\frac{Cost(A_\alpha, x)}{x}\right)$$

It is easy to observe that x must not be placed very close to the origin else the CR will be very large. We will see that $CR \leq 9$ [11] but we do a more general analysis since we will be considering the average-case performance of the same algorithm. Hence we need to know the worst-case performance.

We have the following lemma.

Lemma 2.3. *Let $\alpha > 1$ and $x \in (\alpha^{2l}, \alpha^{2l+2}]$ then $Cost(A_\alpha, x) = 2\frac{(\alpha^{2l+2}-1)}{\alpha-1} + x$.*

Proof. Considering the multiple turn-points for the cow path problem, total cost can be calculated as following when the exit is placed between points α^{2l} and α^{2l+1}

$$\begin{aligned} Cost(A_\alpha, x) &= 2\alpha^0 + 2\alpha^1 + 2\alpha^2 + ... + 2\alpha^{2l} + 2\alpha^{2l+1} + x \\ &= 2\sum_{i=0}^{i=2l+1}(\alpha^i) + x \\ &= 2\frac{(\alpha^{2l+2}-1)}{\alpha-1} + x. \end{aligned}$$

□

Consequently, we conclude with the following theorem

Theorem 2.4. *Let $\alpha > 1$ then $Wrs(A_\alpha) = 2\left(\frac{\alpha^2}{\alpha-1}\right) + 1$.*

Proof. Let $x > 1$ since $\alpha > 1, \exists\ l : x \in (\alpha^{2l}, \alpha^{2l+2}]$. Suppose $x = \alpha^{2l} + \epsilon$.

It is clear that the offline cost x will be $\alpha^{2l} + \epsilon$. Then by lemma 2.3

$$\begin{aligned}\frac{Cost(A_\alpha, x)}{x} &= 2\frac{\alpha^{2l+2} - 1}{(\alpha - 1)(\alpha^{2l} + \epsilon)} + 1 \\ &\leq 2\frac{\alpha^{2l+2} - 1}{(\alpha - 1)(\alpha^{2l})} + 1.\end{aligned}$$

Therefore,

$$\begin{aligned}\sup_{x \geq 1} \frac{Cost(A_\alpha, x)}{x} &\leq \sup_{l \in N} \left(2\frac{\alpha^{2l+2} - 1}{(\alpha - 1)(\alpha^{2l})} + 1\right) \\ &= \lim_{l \to \infty} \left(2\frac{\alpha^{2l+2} - 1}{(\alpha - 1)(\alpha^{2l})} + 1\right) \\ &= 2\frac{\alpha^2}{(\alpha - 1)} + 1.\end{aligned}$$

Therefore, competitive ratio (CR) for algorithm A_α can be written as

$$Wrs(A_\alpha) = 2\left(\frac{\alpha^2}{\alpha - 1}\right) + 1.$$

□

We can conclude the upper bound value of competitive ratio for the cow path problem with the following corollary.

Corollary 2.5. *The best value of A_α is when α=2, then Competitive ratio for A_α is 9.*

Proof. In order to compute the best possible CR, we need to minimize $Wrs(A_\alpha)$, when expansion factor $\alpha > 1$, given by function

$$f(\alpha) = 2\frac{\alpha^2}{(\alpha - 1)} + 1.$$

Therefore,

$$\begin{aligned} f'(\alpha) &= 0 \\ or \quad \frac{4\alpha}{(\alpha-1)} - \frac{2\alpha^2}{(\alpha-1)^2} &= 0 \\ \frac{2\alpha(\alpha-2)}{(\alpha-1)^2} &= 0 \\ or \;\; \alpha = 0 \;\; or \;\; 2 & \end{aligned}$$

Since α can not be 0 so it can only be 2. Now we prove that $a = 2$ is indeed a minimizer.

$$\begin{aligned} f''(\alpha) &= \frac{4}{(\alpha-1)} - \frac{4\alpha}{(\alpha-1)^2} - \frac{4\alpha}{(\alpha-1)^2} + \frac{4\alpha^2}{(\alpha-1)^3} \\ &= -4 - \frac{4\alpha}{(\alpha-1)^2} + \frac{4\alpha^2}{(\alpha-1)^3}. \\ f''(2) &= 4 \end{aligned}$$

Therefore for $\alpha = 2,\ \ f''(\alpha) > 0$.

$$f(\alpha) = \frac{2 \cdot 2^2}{2-1} + 1$$

$$= 8 + 1$$

Therefore,

$$f(2) = 9.$$

By minimizing the function $f(\alpha)$, it proves this algorithm has competitive ratio ≤ 9 at the optimum expansion factor $\alpha = 2$. This confirms the upper bound for a subset of algorithms. □

We will also sketch the proof [11] as to why no deterministic algorithm can have a competitive ratio better than 9. An algorithm for solving cowpath problem is determined by the sequence of turning points $\{a_0, a_1, a_2, ...\}$ satisfying previous properties (see observation 2.1).

Lemma 2.6. *For any sequence* $\{a_0, a_1, a_2, ...\}$ $CR \geq 9$.

Proof. Referring to figure 2.1, we have the sequence of turning points $\{a_0, a_1, a_2, ...\}$ that satisfy the properties previously mentioned (see observation 2.1).

For an exit placed near or after point a_0

$$CR \geq \frac{2a_0 + 2a_1 + a_0}{a_0}.$$

For an exit placed near or after point a_1

$$CR \geq \frac{2a_0 + 2a_1 + 2a_2 + a_1}{a_1}.$$

For an exit placed near or after point a_k

$$CR \geq \frac{2a_0 + 2a_1 + 2a_2 + ... + 2a_k + 2a_{k+1} + a_k}{a_k}.$$

$$g_k = \frac{2\sum_{i=0}^{i=k+1}(a_i)}{a_k} + 1.$$

Where g_k is a lower bound to the competitive ratio for an algorithm where exit is placed after k turning points and a_i expansion factor. So a bound on CR can be given by

$$min\ CR \geq g_k, \quad k = 0, 1, .., \infty.$$

Theorem [11] says that the value of $min\ \ CR$ can not be better than 9. Since g_k has infinite variables $(a_0, a_1, ...)$, so to prove the theorem, we fix k by choosing a variable

l to bound k s.t.

$$min\, CR \geq g_k, \quad k = 0, 1, .., l.$$

Now g_k will have $l+1$ variables $(a_0, a_1, ..., a_l)$. We consider mathematical program to determine the optimal value of CR. Let c_l be the optimal value of CR. One can prove that $\lim_{k\to\infty} CR = 9$ which is technical theorem. Following table shows the result of the mathematical program when we compute various values of c_l, numerically :

l	c_l
0	5
1	6.23
2	7
3	7.49
4	7.82
5	8.06
6	8.23
7	8.36
8	8.46

Table 1: Values of c_l (optimal CR) for the corresponding value of l generated using Mathematica

□

2.2 Average-case analysis

In most cases of search problems, consideration of *average-case performance* of an algorithm in comparison to the *worst-case performance* can help determine efficient usage of the resources. Our goal is to analyze average value of A_α when exit is

placed uniformly at random. Also, because domain is unbounded there could be infinite possible inputs. In order to bound the domain, we introduce a distance D from the origin s.t. $D \to \infty$. Since Average cost is calculated as the expected cost over all the possible inputs in the given range $(1, D]$, so we define Average CR in this interval as

$$E_{x\in(1,D]}\left[\frac{Cost(A_\alpha, x)}{x}\right].$$

Ideally we want to compute average-case CR as $\lim_{D\to\infty} E_{x\in(1,D]}[\frac{Cost(A_\alpha,x)}{x}]$ but analysis shows that the limit does not exist as $D \to \infty$ therefore we take limsup. Hence, the Average CR is defined as

$$Avg(A_\alpha) = \limsup_{D\to\infty} E_{x\in(1,D]}\left[\frac{Cost(A_\alpha, x)}{x}\right].$$

For average-case analysis, we need to find the algorithm A_α that will minimize the expected cost. We introduce λ and D^l_λ s.t. $\lambda \in (0, 1]$ and D^l_λ is a convex combination of α^{2l} and α^{2l+2} s.t.

$$D^l_\lambda = \lambda\alpha^{2l+2} + (1-\lambda)\alpha^{2l}.$$

where $D \in (\alpha^{2l}, \alpha^{2l+2}]$. We will calculate expected cost $E_{x\in(1,D^l_\lambda]}\left[\frac{Cost(A_\alpha,x)}{x}\right]$ (see

lemma 2.7). Then we compute

$$\lim_{l\to\infty} E_{x\in(1,D_\lambda^l]}\left[\frac{Cost(A_\alpha,x)}{x}\right]$$

(see lemma 2.8). Finally we calculate average CR as

$$\sup_{\lambda\in(0,1)} \lim_{l\to\infty} E_{x\in(1,D_\lambda^l]}\left[\frac{Cost(A_\alpha,x)}{x}\right]$$

(see lemma 2.9). Let $f(\lambda,\alpha) = \lim_{l\to\infty} E_{x\in(1,D_\lambda^l]}\left[\frac{Cost(A_\alpha,x)}{x}\right]$. We need to find the λ that maximizes $f(\lambda,\alpha)$. Then we will choose the best α so as to minimize that limsup.

We have the following lemma.

Lemma 2.7. *Expected CR is given as*

$$E_{x\in(1,D_\lambda^l]}\left[\frac{Cost(A_\alpha,x)}{x}\right] = \frac{4\ln\alpha}{(\alpha-1)(D_\lambda^l-1)}\left[\frac{\alpha^2}{\alpha^2-1}(\alpha^{2l}-1)-l\right]+ \frac{2(\alpha^{2l+2}-1)}{(\alpha-1)(D_\lambda^l-1)}\ln\left(D_\lambda^l/\alpha^{2l}\right)+1$$

where $D_\lambda^l = \alpha^{2l}[\lambda\alpha^2+1-\lambda]$

For notational convenience, D_λ^l *is replaced by* D *as analysis holds true when* $D \in (\alpha^{2l},\alpha^{2l+2})$.

Proof. Expected cost

$$
\begin{aligned}
E_{x\in(1,D]}\left[\frac{Cost(A_\alpha,x)}{x}\right] &= \frac{1}{D-1}\int_1^D \frac{Cost(A_\alpha,x)}{x}dx \\
&= \frac{1}{D-1}\left[\sum_{i=0}^{l-1}\left(\int_{\alpha^{2i}}^{\alpha^{2i+2}} \frac{Cost(A_\alpha,x)}{x}dx\right) + \int_{\alpha^{2l}}^{D}\left(\frac{Cost(A_\alpha,x)}{x}\right)dx\right] \\
&= \frac{1}{D-1}\left[\sum_{i=0}^{l-1}\int_{\alpha^{2i}}^{\alpha^{2i+2}}\left(2\frac{\alpha^{2i+2}-1}{(\alpha-1)x}+1\right)dx + \int_{\alpha^{2l}}^{D}\left(2\frac{\alpha^{2l+2}-1}{(\alpha-1)x}+1\right)dx\right] \\
&= \frac{2}{(\alpha-1)(D-1)}\left[\sum_{i=0}^{l-1}(\alpha^{2i+2}-1)\int_{\alpha^{2i}}^{\alpha^{2i+2}}\frac{1}{x}dx + \right. \\
&\qquad \left. (\alpha^{2l+2}-1)\int_{\alpha^{2l}}^{D}\frac{1}{x}dx\right] + 1 \\
&= \frac{2}{(\alpha-1)(D-1)}\left[\sum_{i=0}^{l-1}(\alpha^{2i+2}-1)\ln(\alpha^2) + (\alpha^{2l+2}-1)\ln\left(D/\alpha^{2l}\right)\right] + 1 \\
&= \frac{4\ln\alpha}{(\alpha-1)(D-1)}\left[\frac{\alpha^2}{\alpha^2-1}(\alpha^{2l}-1) - l\right] + \\
&\qquad \frac{2(\alpha^{2l+2}-1)}{(\alpha-1)(D-1)}\ln\left(D/\alpha^{2l}\right) + 1.
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
E_{x\in(1,D]}\left[\frac{Cost(A_\alpha,x)}{x}\right] &= \frac{4\ln\alpha}{(\alpha-1)(D-1)}\left[\frac{\alpha^2}{\alpha^2-1}(\alpha^{2l}-1) - l\right] + \\
&\qquad \frac{2(\alpha^{2l+2}-1)}{(\alpha-1)(D-1)}\ln\left(D/\alpha^{2l}\right) + 1.
\end{aligned}
$$

This will be true $\forall D \in (\alpha^{2l}, \alpha^{2l+2})$ □

Lemma 2.8. *For $\alpha > 1$ and $\lambda \in (0, 1]$, let $D_\lambda^l = \alpha^{2l}[\lambda\alpha^2 + 1 - \lambda]$. Then*

$$\lim_{l\to\infty} E_{x\in(1,D_\lambda^l]}\left[\frac{Cost(A_\alpha, x)}{x}\right] = \frac{2\alpha^2}{(\alpha-1)(\lambda\alpha^2+1-\lambda)}\left[\frac{2\ln\alpha}{\alpha^2-1} + \ln(\lambda\alpha^2+1-\lambda)\right] + 1.$$

Proof. Expected cost by lemma 2.7

$$\begin{aligned}
E_{x\in(1,D_\lambda^l]}\left[\frac{Cost(A_\alpha, x)}{x}\right] &= \frac{4\ln\alpha}{(\alpha-1)(D-1)}\left[\frac{\alpha^2}{\alpha^2-1}(\alpha^{2l}-1) - l\right] \\
&\quad + \frac{2(\alpha^{2l+2}-1)}{(\alpha-1)(D-1)}\ln(D/\alpha^{2l}) + 1. \\
\lim_{l\to\infty} E_{x\in(1,D_\lambda^l]}\left[\frac{Cost(A_\alpha, x)}{x}\right] &= \frac{4\ln\alpha(\alpha^2)}{(\alpha-1)(\alpha^2-1)}\lim_{l\to\infty}\left[\frac{\alpha^{2l}-1}{(D-1)}\right. \\
&\quad \left. + \frac{2}{\alpha-1}\ln(\lambda\alpha^2+1-\lambda)\lim_{l\to\infty}\frac{\alpha^{2l+2}-1}{(D-1)}\right] + 1 \\
&= \frac{4\ln\alpha(\alpha^2)}{(\alpha-1)(\alpha^2-1)}\left[\frac{1}{\lambda\alpha^2+1-\lambda}\right. \\
&\quad \left. + \frac{2}{\alpha-1}\ln(\lambda\alpha^2+1-\lambda)\frac{\alpha^2}{\lambda\alpha^2+1-\lambda}\right] + 1 \\
&= \frac{2\alpha^2}{(\alpha-1)(\lambda\alpha^2+1-\lambda)}\left[\frac{2\ln\alpha}{\alpha^2-1} + \ln(\lambda\alpha^2+1-\lambda)\right] + 1.
\end{aligned}$$

□

Average-case, $g(\alpha)$ calculated as $sup_{(\lambda\in(0,1])}\lim_{l\to\infty} E_{x\in(1,D_\lambda^l)}\left[\frac{Cost(A_\alpha,x)}{x}\right]$ can be given by the following lemma.

Lemma 2.9. *An upper bound to the Average-case CR is* $g(\alpha) = \frac{\left[\exp\left(\frac{\alpha^2-1-2\log\alpha}{\alpha^2-1}\right)-1\right]}{(\alpha^2-1)}$.

Proof. We have Expected CR (by lemma 2.8)

$$\lim_{l\to\infty} E_{x\in(1,D_\lambda^l)}\left[\frac{Cost(A_\alpha,x)}{x}\right] = \left(\frac{2\alpha^2}{\alpha-1}\right)t_\alpha(\lambda)+1,$$

where $t_\alpha(\lambda) = \left(\frac{1}{\lambda\alpha^2+1-\lambda}\right)\left(\frac{2ln(\alpha)}{\alpha^2-1}+\ln(\lambda\alpha^2+1-\lambda)\right)$ and $\lambda\in(0,1]$

$t_\alpha(\lambda)$ is dependent on λ. To find $\sup_{0<\lambda<1} t_\alpha(\lambda)$ we maximize $t_\alpha(\lambda)$ and calculate $t'_\alpha(\lambda)=0$ (which can be thought as an adversarial choice).

We know that

$$t_\alpha(\lambda) = \left(\frac{1}{\lambda\alpha^2+1-\lambda}\right)\left(\frac{2ln(\alpha)}{\alpha^2-1}+\ln(\lambda\alpha^2+1-\lambda)\right)$$

Using Mathematica computation, we get the following λ_α when $t_\alpha(\lambda)$ is maximized

$$\lambda_\alpha = \frac{\left[\exp\left(\frac{\alpha^2-1-2\log\alpha}{\alpha^2-1}\right)-1\right]}{(\alpha^2-1)}$$

where $\lambda_\alpha\in[0,1)\ \forall\alpha>1$ as shown in Figure 2.2. Then,

$$t_\alpha(\lambda_\alpha) = \frac{\alpha^{\left(\frac{2}{\alpha^2-1}\right)}}{e}.$$

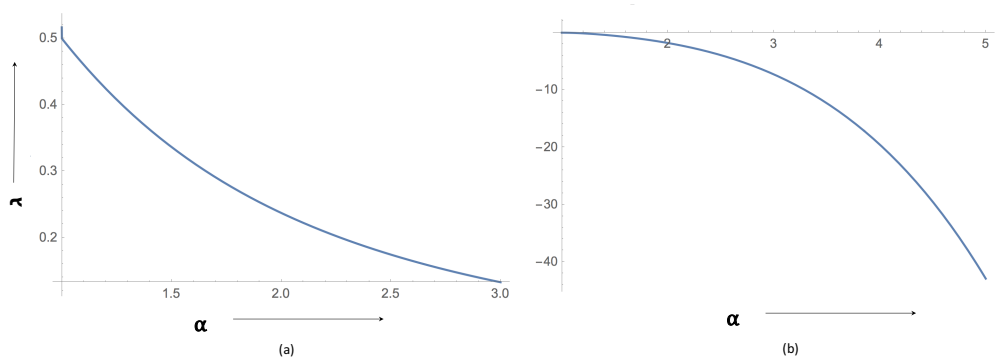

Figure 2.2: (a): Graph between α and λ shows that α_λ is optimum close to *approx* 0.5. (b) Curve confirms that λ_α is a maxima.

Therefore expected CR becomes,

$$g_\alpha(\lambda_\alpha) = \left(\frac{2\alpha^2}{\alpha - 1}\right) \frac{\alpha^{\left(\frac{2}{\alpha^2-1}\right)}}{e} + 1$$

where $\lambda \in (0, 1] \; \forall \; \alpha > 1$.

The value is maximized at λ_α, if $t''_\alpha(\lambda_\alpha) < 0 \; \forall \; \alpha > 1$. Using symbolic calculations from Mathematica computation, we plot the following two graphs (Figure 2.2). Figure 2.2b proves that $g_\alpha(\lambda_\alpha)$ has a local maximum at λ_α.

$t_\alpha(\lambda_\alpha)$ is entirely a function of α. One can verify that $\lambda_\alpha \in [0, 1] \; \forall \; \alpha > 1$. Figure 2.2a plots λ_α for the varying values of α.

□

Observation 2.10. The best value of α is 2.70453 at 5.32685 expected evacuation time.

Using Mathematica computation, we obtain the graph (Figure 2.3) between

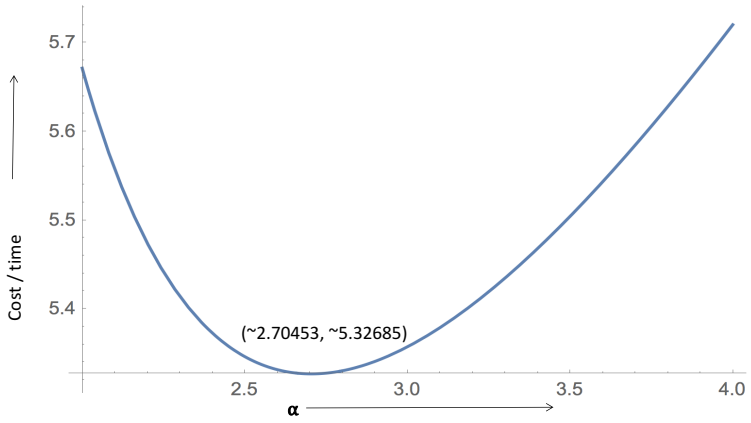

Figure 2.3: Plot of expected cost (time) against expansion factor α showing the initially cost decreases until ≈ 2.7 then it starts to increase.

the values of α and the corresponding cost (in terms of time). We see that the cost initially is reducing as the α increases but after a certain point cost will start to increase. The best value of α is close to ≈ 2.70453, which can be calculated numerically by minimizing $f(\alpha)$ so $f'(\alpha) = 0$ and the corresponding CR to locate the exit is ≈ 5.32685.

2.3 Trade off - Worst-case vs Average-case

Upon completing various relevant calculations, the following parametric curve (Figure 2.4a) gives us a visual representation of the trade offs between the costs for the Average-case ($g(\alpha)$) and the Worst-case ($f(\alpha)$) scenarios. The curve is drawn between $(f(\alpha), g(\alpha))$ where f is the worst-case performance of algorithm A_α and g is the average-case performance of A_α. When the minimum worst-case value of $f(\alpha)$

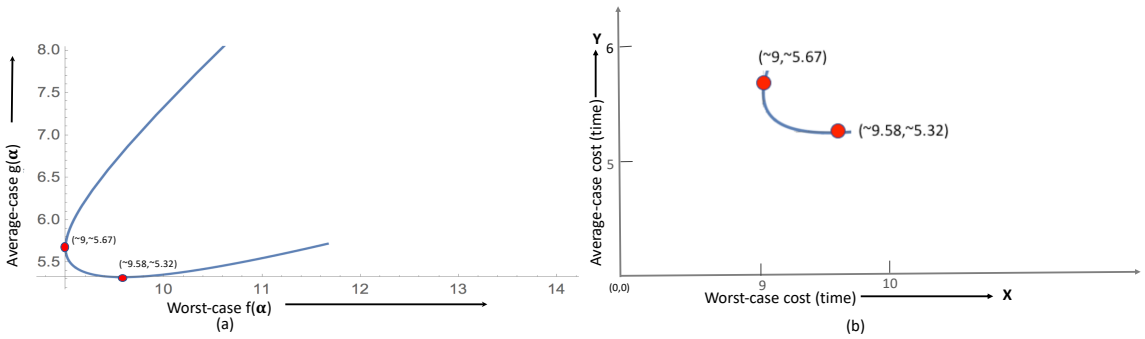

Figure 2.4: (a): Worst-case vs Average-case Trade off; (b): Critical points between Average-case and Worst-case for the Cow path problem.

is ≈ 9 then the average-case is ≈ 5.67. When we try to minimize the average-case value at ≈ 5.32 then the worst-case is increased to ≈ 9.58. Notice the curve between the points $(\approx 9, \approx 5.67)$ and $(\approx 9.58, \approx 5.35)$ (Figure 2.4b). There is a delicate balance of the efficient values between these two points. As the worst-case performance cost worsens from ≈ 9 to ≈ 9.58, the average-case performance cost improves from ≈ 5.67 to ≈ 5.32. Similarly when average-case performance cost worsens from ≈ 5.32 to ≈ 5.67 then the worst- case performance improves from ≈ 9 to ≈ 9.58. These points are crucial from the efficiency perspective otherwise there are more than one corresponding value for each f and g. We notice that those points fall outside the range of these crucial points. For example, for worst-case CR 10 there are two values for average-case CR ≈ 5.4 and ≈ 7. Its obvious that 5.4 is more efficient but both the values of f and g outside the crucial points proving that our range covers all the efficient combinations of worst-case and average-case scenarios. It is clear that when we try to minimize $f(\alpha)$ then there is little control over $g(\alpha)$

and vice versa. Another important consideration at this point is the availability of the resources (time, in this case). While considering worst-case, it is assumed that we have unlimited resource available to be able to locate the exit but in reality, consideration of limited resources is crucial as we may have limited time to evacuate or limited gas in the vehicle to be able to search an area. We observed this trade-off phenomenon for the Cowpath problem and it will be very interesting to see how this relationship changes for other algorithms, for other problems.

Chapter 3

EVACUATION PROBLEM ON A DISC

This section first describes a specific evacuation problem which is followed by the relevant evacuation algorithms and their worst-case scenarios that have been studied so far. Then we calculate average-case CR for them and compare the average-case CR with the worst-case CR values. By understanding the specific phases of the algorithms, that contribute positively and negatively to the CR, we introduce a new algorithm that has best of the both worlds. Analysis of this new algorithm follows afterwards with respect to its worst-case and average-case performance.

Our problem comes under the umbrella of 'Evacuation problems' in the field of mobile agent computing. The algorithm and the analysis in the previous section (Cowpath problem) will help to lay the groundwork for our specific problem.

3.1 Problem Definition

We consider two mobile search agents, robots R_1 and R_2, who start from the centre O, of a *unit disk*. They are trying to locate a hidden exit and evacuate, in the least possible time, determined only when the last robot has evacuated. The stationary exit is located somewhere on the periphery of this disc and its location is unknown to the robots. The exit is discovered only when the robot is at the same coordinates

as the exit. Robots can move anywhere on the disc with a *maximal speed 1.* Robots can not see each other. In order to communicate information to each other, they have to be at the same location. This is called *face-to-face (F2F)* communication. When the last robot reaches the exit that is the *Evacuation time* for that algorithm. The trajectories of both the robots are determined prior to the start and both know the trajectories of one another. Therefore, at any given time each robot is aware of the location of the other robot.

Similar to the Cowpath problem, the search strategy (algorithm) is announced first and then the adversary determines the location of the exit. On one hand when we consider the worst-case cost, the adversary has the power to choose the worst-possible exit after searcher announces a deterministic search strategy but when a randomized search strategy is announced adversary's power to choose worst-possible exit diminishes because searcher's starting point will be arbitrary as a point x chosen uniformly at random on a unit circle ($x \in (0, 2\pi)$) which assumes that there are unlimited number of random bits. On the other hand when we consider average-case cost, for a deterministic strategy chosen by the searcher, no matter which point adversary chooses (uniformly at random) the algorithm followed by the searcher will achieve the same average-case cost. It is due to the unique nature of our problem on a disc that adversary has no power to choose a worst exit in cases of the randomized algorithm with deterministic input or deterministic algorithm with randomized input. It is important to mention another unique situation in

this evacuation problem of the two robots on a unit disc. For our problem, the competitive ratio will be same as the worst case cost due to the unit disc.

In the field of computer science, efforts have been to solve the search problem assuming the availability of unlimited time-resource. We may have unlimited time but other energy resources such fuel available to run a ship or plane for evacuation will be limited. For example, while doing rescue operation in the sea, the ship may have limited capacity in the fuel tank that will limit search due to the unavailability of the fuel after a certain point. If searchers do not have the resources to achieve the worst-case scenario, we would like to consider the average-case scenario where the possibility of achieving the goal of evacuation is still high despite the limited resources therefore, we will analyze the evacuation costs of the worst-case scenario and the worst average-case scenario for a randomized algorithm considering limited resources. We continue to determine the cost in terms of time spent and the efficiency of the algorithms to locate and evacuate.

A *feasible algorithm* is determined by the trajectories of the robots, in which both robots evacuate eventually through the exit. If $R_1(t)$ and $R_2(t)$ represent the location of the robots R_1, and R_2 at time t then we say that trajectories $R_1(t), R_2(t)$ are feasible if

- $R_1(0) = R_2(0)$
- Trajectories $R_1(t), R_2(t)$ have speed ≤ 1
- There is some time $t_0 > 1$, such that each point of the unit circle is visited

(searched) by at least one robot in the time window $[0, t_0]$. We refer to the smallest t_0 as the search time of the circle.

- At the exit, robots will be at the same location at time t', $R_1(t') = R_2(t')$ eventually

The performance of the algorithm is determined by how fast can it evacuate all the robots through the exit.

3.2 Terminology

The centre O of the disc has the coordinates (0,0) in the Cartesian plane. Any point on the perimeter of the circle can be represented by $\big(cos(x), sin(x)\big), x \in (0, 2\pi]$. For the ease of presentation we will denote it as $cycle(x)$. $\mathcal{A}$ represents an evacuation algorithm. The cost $C(x)$ incurred by an algorithm is considered as the time taken to evacuate both the robots when exit is at $cycle(x)$. The searcher can either follow a pre-determined algorithm or randomly choose an algorithm over another probabilistically. The Average-case cost is calculated as the expected cost $E[C(x)]$ when exit is placed uniformly at random.

Since we are concerned with the performance of the algorithm, to determine the tradeoffs between the worst-case and the average-case costs, so we define *Efficiency* of the algorithm $\mathcal{A}$ as $\Big(Avg(\mathcal{A}), Wrs(\mathcal{A})\Big)$ where:

$$Avg(\mathcal{A}) := E_{x\in[0,2\pi)}[C(x)],$$

$$Wrs(\mathcal{A}) := \sup_{x\in[0,2\pi)}[C(x)].$$

We introduce a value ω to bound the worst-case performance of an algorithm $\mathcal{A}$ such that the worst-case performance will not exceed ω. Then we consider problem

$$\min \frac{1}{2\pi} \int_0^{2\pi} C(x)dx \text{ s.t. } C(x) \leq \omega, \forall x \in [0, 2\pi)$$

where $C(x)$ is the total cost. In the latter part, we will see that problem is interesting when we consider the performance of the algorithm between two specific points (ω_1 & ω_2) on the efficiency graph, as the other values become trivial when considering efficiency of the algorithm.

3.3 Evacuation Time

For the ease of calculations and analysis, we will consider the entire operation to be divided into *search phase S(x)* and *evacuation phase* $\varepsilon(x)$. *S(x)* is the time spent in searching for the exit, located at a point x and $\varepsilon(x)$ is the time spent in locating the other robot. Since both the robots will be travelling back together to the exit point so $\varepsilon(x)$ will be doubled. Therefore, the total cost for an algorithm *C(x)* to locate an exit and then evacuate both the robots is calculated by adding the search time (*S(x)*) to evacuation time ($2\varepsilon(x)$). Therefore cost,

$$C(x) = 1 + S(x) + 2\varepsilon(x).$$

It is worth noting that *C(x)* represents cost of any algorithm. In an effort to simplify notation, we will determine the trajectories of the robots from the moment the two

robots reach the perimeter of the disc until the disc is searched & robots exited.

Therefore, worst-case cost,

$$\begin{aligned} Wrs\big(C(x)\big) &= \sup_{x\in[0,2\pi)} C(x) \\ &= 1 + \sup_{x\in[0,2\pi)} \Big(S(x) + 2\varepsilon(x)\Big). \end{aligned} \tag{3.1}$$

For average case cost,

$$\begin{aligned} Avg\big(C(x)\big) &= \frac{1}{2\pi}\int_0^{2\pi} C(x)dx \\ &= 1 + \frac{1}{2\pi}\int_0^{2\pi} \Big(S(x) + 2\varepsilon(x)\Big)dx. \end{aligned} \tag{3.2}$$

We have the following lemma.

Lemma 3.1. *Suppose exit is located at cycle*(x) *and Robot* R_1 *finds the exit first, by following the algorithm* $\mathcal{A}$*. Then* $\varepsilon(x) = \bar{t} - S(x)$ *where* $\bar{t}(x)$ *is the smallest root,* $\bar{t} \geq S(x)$*, of function*

$$k(x,t) := ||R_2(t) - R_1(S(x))|| - t + S(x)$$

Proof. We will prove that $k(x,t)$ has roots and $\bar{t}(x) \geq S(x)$.

$k(x,t)$ is a continuous function, defined at time t. For fixed x, $k : \mathbb{R}_+ \to \mathbb{R}$. When exit is located, $t = S(x)$ and both robots are not at the same location ($R_2(S(x)) \neq$

$R_1(S(x))$, then

$$k(x, S(x)) := \underbrace{||R_2(S(x)) - R_1(S(x))||}_{} - \underbrace{S(x) + S(x)}_{}$$

$$:= \qquad > 0 \qquad\qquad 0$$

$$k(x, S(x)) > 0$$

Therefore, $k(x, S(x))$ will be positive. $k(x, S(x)) = 0$ when both robots are moving together and locate the exit then $R_2(S(x)) = R_1(S(x)$.

After a large enough time t' where R_1 and R_2 arrive together at exit,

$$k(x, t') := \underbrace{||R_2(t') - R_1(S(x))||}_{} - \underbrace{t'}_{} \quad + \underbrace{S(x)}_{}$$

$$:= \qquad 0 \qquad - large\, t' \quad + fixed\, S(x)$$

$$k(x, t') < 0$$

By mean value theorem, $\exists \bar{t} > S(x) : k(x, \bar{t}) = 0$. We choose smallest $\bar{t} \geq S(x)$. Now we show that $\varepsilon(x) = \bar{t} - S(x)$.

We assume R_1 locates the exit (Figure 3.1). Therefore, at the search time $S(x)$, robots R_1 and R_2 will be at $R_1(S(x))$ and $R_2(S(x))$ respectively. R_1 then locates R_2 via the shortest route at point T after $\bar{t}$. Then both return to exit. Therefore at $\bar{t}$,

$$R_1(\bar{t}) = R_2(\bar{t})$$

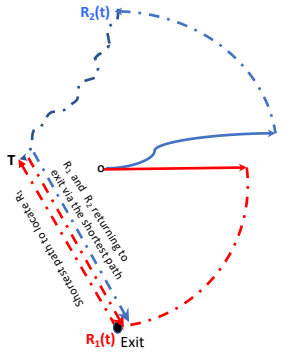

Figure 3.1: The distance, travelled by R_1 and R_2 after time t, will be the same and since both are moving at the same speed so the time taken by both the robots to travel to a point T will also be the same.

Since R_1 and R_2 move at unit speed, $\|R_2(\bar{t}) - R_1(S(x))\| = \bar{t} - S(x)$. Clearly, evacuation time $\varepsilon(x) = \bar{t} - S(x)$ or $\varepsilon(x) = \|R_2(\bar{t}) - R_1(S(x))\|$. □

This theorem is useful as it provides us with a general equation that can be applied widely for any search spaces.

3.4 Benchmark Algorithms

In this section we provide brief overview of some benchmark algorithms that have helped us lay the ground to explore new algorithms with improved performance. $\mathcal{B}_1$ and $\mathcal{B}_2$ are two benchmark algorithms with worst-case costs ≈ 7.28 & ≈ 5.740 respectively as well as average-case costs ≈ 4.14 & ≈ 5.1172 respectively.

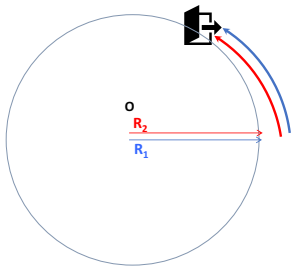

Figure 3.2: Naive algorithm - R_1 and R_2 start at origin O then arriving on periphery, both move together in one direction until the exit is found. Once the exit is located, both evacuate together.

3.4.1 Benchmark Algorithm $\mathcal{B}_1$ - Naive

In this benchmark algorithm, robots follow a very simple algorithm to reach the exit and to evacuate. Robots R_1 and R_2 start from the centre of the disc towards the periphery (Figure 3.2). Once on the periphery both move together in the same direction. After the exit is located both evacuate together.

Mathematically, $\mathcal{B}_1$ can be represented as

$$\varepsilon(x) = 0$$

$$S(x) = x$$

$$\forall t \in (0, 2\pi], R_1(t) = R_2(t) = \text{cycle}(t)$$

From equation 3.1, worst-case cost,

$$\begin{aligned} Wrs(\mathcal{B}_1) &= \sup_{x\in[0,2\pi)} C(x) \\ &= 1 + \sup_{x\in[0,2\pi)} \Big(S(x) + 2\varepsilon(x) \Big) \\ &= 1 + \sup_{x\in[0,2\pi)} S(x) \qquad (note : \varepsilon(x) = 0) \\ &= 1 + \sup_{x\in[0,2\pi)} x \\ &= 1 + 2\pi \\ &\approx 7.28 \end{aligned}$$

So, in the worst-case scenario, exit is found almost at the end on the periphery. Therefore, $Wrs(\mathcal{B}_1) \approx 7.28$. From equation 3.2, average-case cost,

$$\begin{aligned} Avg(\mathcal{B}_1) &= \frac{1}{2\pi} \int_0^{2\pi} C(x)dx \\ &= 1 + \frac{1}{2\pi} \int_0^{2\pi} \Big(S(x) + 2\varepsilon(x) \Big) dx \\ &= 1 + \frac{1}{2\pi} \int_0^{2\pi} x dx \\ &= 1 + \pi \\ &\approx 4.14 \end{aligned}$$

Average-case, $Avg(\mathcal{B}_1) \approx 4.14$. Hence the Efficiency for algorithm $\mathcal{B}_1$ is represnted as $(1 + \pi, 1 + 2\pi)$.

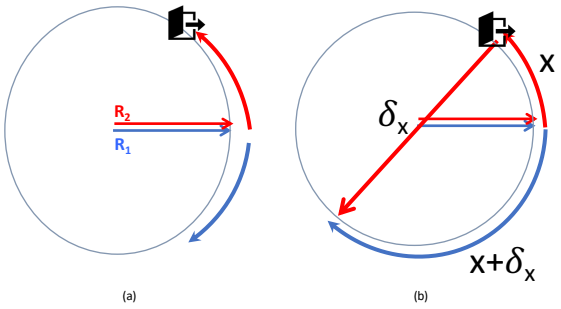

Figure 3.3: Opposite Direction algorithm - R_1 and R_2 start at origin O then arriving on periphery, both move together in opposite directions until the exit is found. Once the exit is located by one robot, she meets with other robot then, both evacuate together.

3.4.2 Benchmark Algorithm $\mathcal{B}_2$ - Opposite

Czyzowicz et al.[29] were the first to introduce the unit disc evacuation algorithm for two robots (Figure 3.3). Here robots R_1 and R_2 start from the centre of the disc towards the periphery. Once on the periphery, both move in the opposite direction. When the exit is located by one robot then she locates the other robot via the most efficient path/chord. Then both robots return to the exit by the same chord distance and evacuate. The cost of the algorithm is parameterized by the location of the exit x, on the perimeter from the starting position. It is easy to observe that for each $t \in (0, 2\pi]$, trajectories $R_1(t)$ and $R_2(t)$ are symmetric with respect to the horizontal axis so it will take a maximum of π time to search the disc on periphery. As we know that the distance covered $(x + \delta_x)$ by both the robots until they meet will be same so chord δ_x is a function of x and it equals the roots of

$$\delta_x = 2sin(\tfrac{2x+\delta_x}{2}).$$

This would be the same result as from Theorem 3.1. In this case we do not use the theorem to compute since it is far more easier through geometric theorem. For any x, $\varepsilon(x) = \delta_x$ and $S(x) = x$ therefore, $C(x) = 1 + x + 2\delta_x$.

We use Mathematica calculations to calculate worst-case performance for algorithm $\mathcal{B}_2$,

$$\begin{aligned} Wrs(\mathcal{B}_2) &= \sup_{x\in[0,\pi]} C(x) \\ &= \sup_{x\in[0,\pi]} (1 + x + 2\delta_x) \approx (5.740) \quad [29] \end{aligned}$$

We also calculate the average-case performance, using Mathematica calculations

$$Avg(\mathcal{B}_2) = E_{x\in[0,\pi]}C(x) = \frac{1}{\pi}\int_0^{\pi}(1 + x + 2\delta_x)dx \approx 5.1172.$$

The worst-case performance of the algorithm occurs when $x \approx 0.968$ (Figure 3.4). Therefore, the *Efficiency* for $\mathcal{B}_2$ is $(5.1172, 5.740)$.

3.4.3 Benchmark Algorithms $\mathcal{B}_{2.1}$ and $\mathcal{B}_{2.2}$

Algorithms $\mathcal{B}_{2.1}$ and $\mathcal{B}_{2.2}$ use $\mathcal{B}_2$ as the base algorithm to evacuate two robots from the unit disc in a F2F model and they provided further enhanced results as compared to the base algorithm. Czyzowicz et. al [36] introduced two new algorithms with forced detour and improved both the lower and upper bounds of the evacuation times. While we briefly mentioned both the algorithms in the literature review

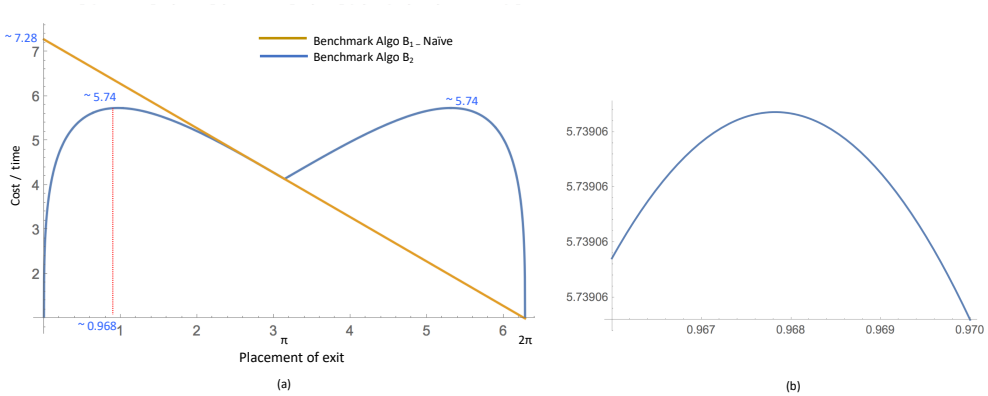

Figure 3.4: (a) Comparing the worst and average case performance of Benchmark algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$; (b) Detailing the critical point ≈ 0.968 when cost is the worst (≈ 5.74).

section, we will be focusing on $\mathcal{B}_{2.1}$ algorithm (Figure 3.5a) in this section as it is closely related to our problem. $\mathcal{B}_{2.1}$ algorithm improved the worst-case performance to ≈ 5.644. It was achieved by introducing a detour as a part of the algorithm that forced detour for the robots after a certain time when the exit is not found. There are two points of significance, B and C in the algorithm. B is the point on periphery where the robot starts the detour and C represents the location of the detour point on the disc, before the robot returns back to the periphery. So after a pre-determined time robot turns in to follow the detour at point B ($B \rightarrow C$) and then it returns back to the periphery via the same path ($C \rightarrow B$). Each set of B and C represents a family of algorithms denoted by $C(\chi, \phi)$ where χ is the distance travelled by the robot on the perimeter, ϕ is the angle at which the detour starts. In their optimal evacuation algorithm, the two robots are forced to meet after visiting a subset of vertices of the disc, even if an exit has not been found at that time.

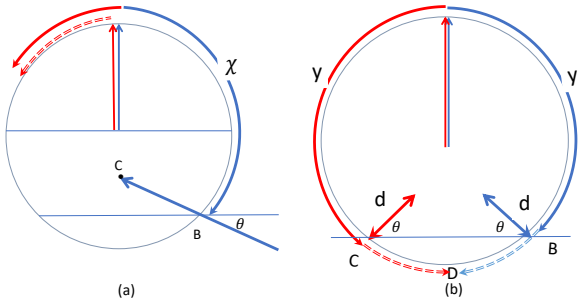

Figure 3.5: Algorithms showing the trajectories of the robots R_1 and R_2 with forced detour - (a) With one forced meeting detour- $\mathcal{B}_{2.1}$; (b) With multiple forced detours -$\mathcal{B}_{2.2}$.

An optimum CR value ≈ 5.644 was achieved for the specific values of $\chi = 2.62359$, $\phi = 0$.

Brandt et al [22] (Figure 3.5b) further improved the upper bound to ≈ 5.625. We refer to the latter algorithm as $\mathcal{B}_{2.3}$. The worst-case evacuation time was improved by forcing multiple detours for both the robots.

Therefore,

$$Wrs(\mathcal{B}_{2.1}) \approx 5.644 \text{ [36] } (when\ \chi = 2.62359,\ and \phi = 0)$$

$$Wrs(\mathcal{B}_{2.2}) \approx 5.625 \text{ [22]}$$

3.4.4 Cost and Efficiency Comparison for $\mathcal{B}_1$ and $\mathcal{B}_2$

Table 2 summarizes the average and worst-case costs for both algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$:

	Cost: Average-case	Cost: Worst-case
Benchmark Algorithm $\mathcal{B}_1$	$1+\pi(\approx 4.14)$	$1+2\pi(\approx 7.28)$
Benchmark Algorithm $\mathcal{B}_2$	≈ 5.12	≈ 5.74

Table 2: Values of average-case and worst-case for $\mathcal{B}_1$ and $\mathcal{B}_2$ benchmark algorithms.

We plot (Figure 3.6) the costs for both the algorithms $\mathcal{B}_1$and $\mathcal{B}_2$ together. We notice that the curve for $\mathcal{B}_2$ is symmetric because when the robots are moving in the opposite directions then the exit will be located when $x \in (0, \pi]$ and the worst-cost (≈ 5.74) is observed at $x \approx 0.968$. We also notice that while $\mathcal{B}_1$ performs very well in the earlier part of the algorithm but the cost-rise is significant towards the later part of the algorithm. Yet the average value looks relatively good due to large variation in the cost over the entire process. In contrast to $\mathcal{B}_1$, $\mathcal{B}_2$ has a higher cost in the earlier part of the algorithm and it varies through the entire process. As a result, the worst performance of $\mathcal{B}_2$ is better than $\mathcal{B}_1$ but the average cost of $\mathcal{B}_1$ is better than $\mathcal{B}_2$.

We previously discussed that our intention is to consider the average-case scenario where the possibility of achieving the goal of evacuation is still high despite the limited resources. So we did a rough representation of the Efficiency Comparison for benchmark algorithms $\mathcal{B}_1$and $\mathcal{B}_2$ (Figure 3.7a). Similar to the Cowpath example earlier, we notice that when the worst-case performance cost improves then the average-case performance cost increases and vice versa. It is not uncommon for many problems to have the lower and upper bounds to be only available numeri-

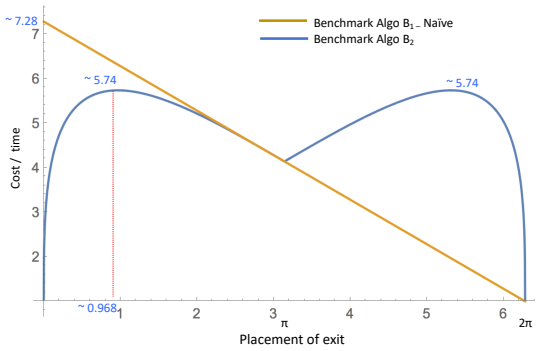

Figure 3.6: Comparing the worst and average case performance of Benchmark algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$ for varying position of exit

cally. For evacuation problems with two robots having F2F communication and a maximum of ω worst-case time also, the cost of best-solution does not exist because at the point where worst-case cost is low, the average-case is high and vice versa. Between these two points, M and O, on the plot (Figure 3.7a), the 'best solution' will depend on the searcher's objective i.e. if she wants low average cost then she will choose a point M or a point in its vicinity while, if she needs worst-case to be low then she will choose point O or a point in its vicinity. This gives us a motivation to explore the algorithms for which efficiency would fall in the vicinity of points M and O, ideally close to line MO.

It is also important to mention the results (Figure 3.7b) of a performance analysis [26] recently done for benchmark algorithms $\mathcal{B}_2$. Figure 3.7b shows a consolidated efficiency-plot for various algorithms as we consider various combinations of χ and ϕ for varying detour-point C as shown in Figure 3.5 where C is the last point located on the disc to end the detour before robot's return back to the periphery. The plot

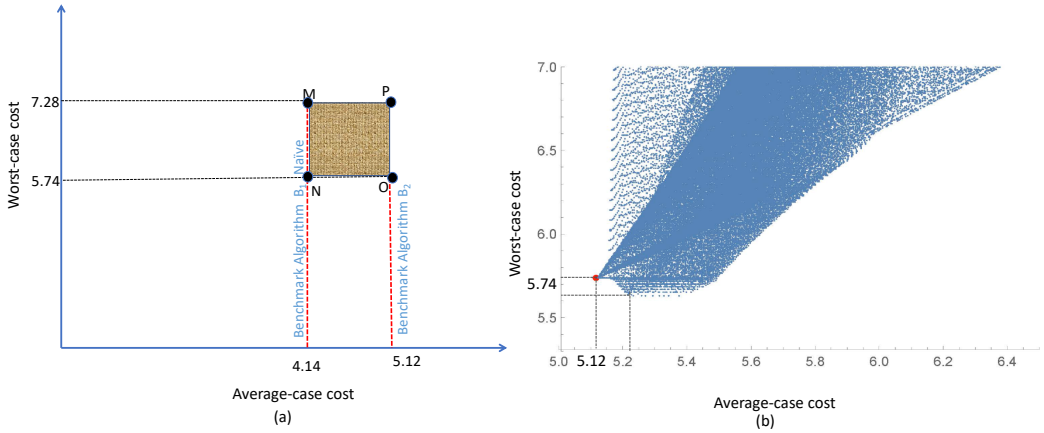

Figure 3.7: Efficiency Comparison: (a) benchmark algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$; (b) A consolidated efficiency plot [26] for families of $\mathcal{B}_2.1$ algorithm with varying values of α and the detour-point C

depicts more than 500,000 different parameter values using lemma 3.1. Since the red point represents the least average ≈ 5.12 for a corresponding worst-case cost ≈ 5.74 which is same as our $\mathcal{B}_2$ therefore they concluded that no other algorithm has an average better than 5.12.

In the next section, we will explore a new algorithm.

3.5 New Algorithm

In an effort to find a possible solution that will help us cover the current gap between points M and O (Figure 3.7), we will explore a new algorithm. We will further analyze to see if this algorithm has an efficient worst-case as well as average-case performance as compared to the benchmark algorithms. We will refer to this new algorithm as $\mathcal{B}_3(\alpha)$, parametrized by the distance α where detour takes place, $\alpha \in (0, \pi]$.

For an algorithm $\mathcal{A}$, efficiency is defined as (Avg($\mathcal{A}$), Wrs($\mathcal{A}$)). For a fixed $\omega \in [5.74, 1+2\pi]$, we restrict our attention to $Wrs(\mathcal{A}) \leq \omega$. Among them we would like to minimize Avg($\mathcal{A}$). Since our goal is to devise an algorithm parametrized by α, so we introduce a family of algorithms $\mathcal{B}_3(\alpha)$ where $\alpha \in (0, \pi]$. When $\alpha = 0$ then $\mathcal{B}_3(0)$ is the benchmark algorithm $\mathcal{B}_1$ and when $\alpha = \pi$ then $\mathcal{B}_3(\pi)$ is the benchmark algorithm $\mathcal{B}_2$. In general, we will find functions f and g such that $Avg(\mathcal{B}_3(\alpha)) = g(\alpha)$ & $Wrs(\mathcal{B}_3(\alpha)) = f(\alpha)$. Given any $\omega \in [5.74, 1+2\pi]$, we will find $f(\alpha) \leq \omega$ and so as to min $g(\alpha)$. While we are able to determine $f(\alpha)$ exactly (closed formula) but $g(\alpha)$ will be computed only numerically.

This way we propose one solution to the optimization problem for an algorithm $\mathcal{A}$ with $min\ Avg(\mathcal{A})$ such that

$$Wrs(\mathcal{A}) \leq \omega \ \ where \ \ \omega \in [5.74, 1+2\pi]$$

We will find $\mathcal{B}_3(\alpha)$ for $\mathcal{A}$ where $\alpha \in (0, \pi]$. It may/not be the optimal solution.

3.5.1 Trajectories of R_1 and R_2 in algorithm $\mathcal{B}_3$

R_1 and R_2 start together from the centre of the disc towards the periphery. Once on the periphery both move in the opposite direction. When the exit is located by one robot then she locates the other robot via the most efficient path/cord. Both return to the exit by the same chord distance and then both evacuate. In case the exit is not located until time α then R_2 turns inwards on chord and follows δ_α to

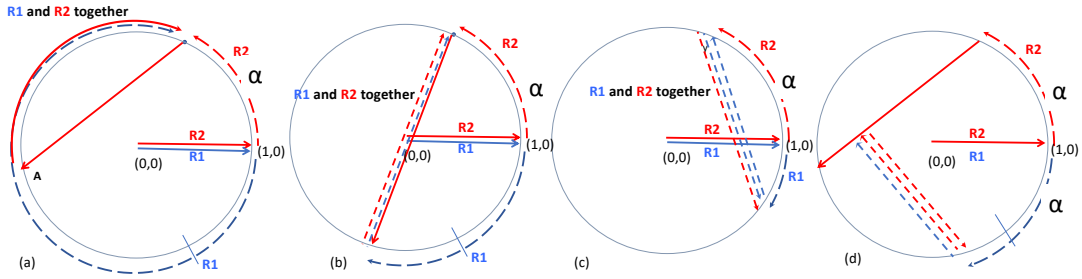

Figure 3.8: R_1 and R_2 trajectories, (a) the exit is found in the latter part of the algorithm when they move together; (b) the exit is found by R_2; (c) the exit is found by R_1; (d) the exit is found by R_1 after R_2 has turned in to catch R_1.

meet R_1 on the other side at the periphery. Then both robots move together, on the part of the periphery that is not yet been searched, until the exit is located. Once the exit is located then both evacuate together.

Robot $\mathbf{R}_1$: go to $cycle(0)$

Search clockwise until $cycle(0)$

Robot $\mathbf{R}_2$: go to $cycle(0)$

Search counter clockwise until $cycle(\alpha)$

Move along line segment $cycle(\alpha)$ until $cycle(-\alpha - \delta_\alpha)$

Continue search clockwise until $cycle(0)$

By definition of function δ_α, two robots meet at time $(1 + \alpha + \delta_\alpha)$.

When exit is located after R_1 and R_2 have met at point A, then both travel together (Figure 3.8a). When the exit is located by R_2 then R_2 will travel to locate R_1 (Figure 3.8b) then both travel back together. When the exit is located by R_1 then R_1 will travel to locate R_2 (Figure 3.8c) then both travel back together. Lastly

when the exit is located by R_1 after α then R_1 will travel towards the R_2 and will catch R_1 (Figure 3.8d) somewhere on the chord then both travel back together.

When considering the extreme values for α, we notice an interesting behaviour of the new algorithm. When $\alpha = 0$ then both the robots are together and $\mathcal{B}_3$ behaves like the naive algorithm $\mathcal{B}_1$. When $\alpha = \pi$ then $\mathcal{B}_3$ behaves like the $\mathcal{B}_2$ algorithm. We will discuss the behaviours of the algorithm as it gets closer to the extreme case in the latter part of this section.

3.5.2 Cost Calculation

For calculating worst-case or average-case performances, we need to find the total cost

$$C(x) = 1 + S(x) + 2\varepsilon(x) \qquad \forall x \in [0, 2\pi).$$

In order to simplify the calculations for $\mathcal{B}_3$, we have subdivided the entire search into various cases based upon the location of the possible exit (Figure 3.9). After completing the calculations we put them all together to determine the worst-case and average-case performances.

When exit is at $cycle(x)$ then $x \in [0, 2\pi)$ and the value of $C(x)$ will depend on the values of $S(x)$ and $\varepsilon(x)$. 3 main cases could result from the different positions of the exit x. In Case 1, exit is located at $cycle(x)$ where $x \in [0, \alpha]$ and robots will follow Figure 3.8b trajectories. In Case 2, exit is located at $cycle(x)$ where

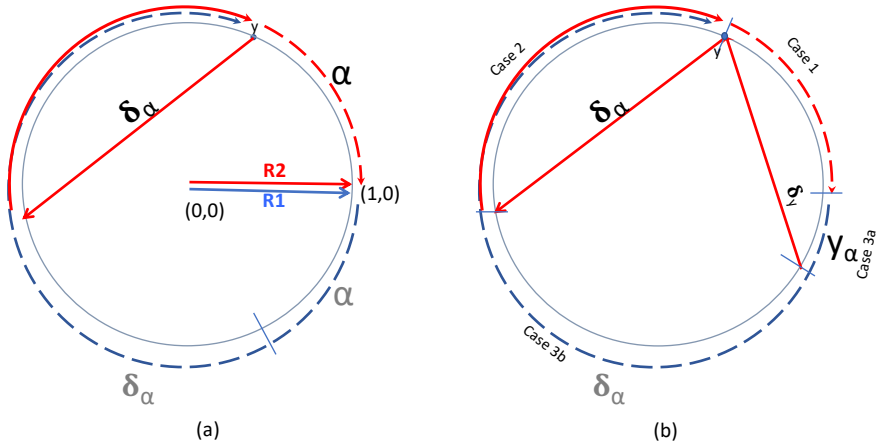

Figure 3.9: (a) Algorithm followed by R_1 and R_2; (b) 3 Cases based upon the location of the exit.

$x \in (\alpha, 2\pi - \alpha - \delta_\alpha]$ and robots will follow Figure 3.8a. In Case 3, exit is located by robot R_1 and it is subdivided into 3a and 3b (Figure 3.9b, see also Figure 3.8c & d). Note that for small enough ϵ, R_1 has time to catch R_2 while it is still on the periphery, when exit is at $cycle(-\epsilon)$. Case 3a includes all these locations of the exit between the points 2π and the last point where R_1 will be able to catch R_2 while it is still on the periphery. Case 3b contains the rest of the points. Here R_2 has already started the inwards-detour. As a result, R_1 and R_2 will meet somewhere on segment AB. Therefore, Case 3b will include all the points between $2\pi - \alpha - \delta_\alpha$ and earliest point where R_1 will be able to catch R_2 on segment AB.

Case 1: When the exit x is located in the range $[0, \alpha]$ (Figure 3.9b) and it is searched by R_2 then

Search time $S(x) = x$

δ_α is the length of the chord traveled by R_2 after turning inwards for the detour,

until it reaches the periphery of the disc on the other side. When the exit is at x then the distance travelled on the chord will be δ_x. Therefore,

Evacuation time $\varepsilon(x) = \delta_x$.

Therefore, $\boxed{C(x) = 1 + x + 2\delta_x}$

Of all the possible values of α there is a point α_0, where $C(x)$ has the maximum value (Figure 3.6) so,

$$\sup_{x\in(0,\alpha]} C(x) = \begin{cases} C(\alpha) & \alpha \leq \alpha_0 \\ C(\alpha_0) & \alpha > \alpha_0 \end{cases}$$

and $C(\alpha_0) \approx 5.74$ for $\alpha_0 \approx 0.968$. From 0 to α_0 the cost increases steadily as the α increases but after α_0 cost starts to decrease as the total cost (search & evacuation) cost reaches its max due to the longest possible path covered by the robots. After α_0, we see a decrease in the cost despite increasing the value of α. It happens because algorithm has achieved the maximum cost due to the the longest possible path been covered. The total length of the search path will only reduce after α_0 as robots are moving towards each other, resulting in the reduced time to catch the other robot.

Case 2: Exit x is located in the range $(\alpha, 2\pi - \alpha - \delta_\alpha]$. As a result, the exit will be located by both the robots together. Hence

$\varepsilon(x) = 0$.

$S(x) = 2\pi - x$.

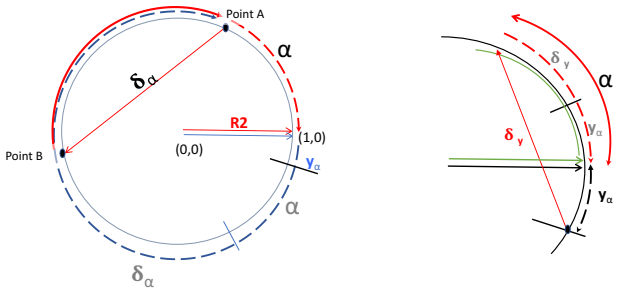

Figure 3.10: (a) Case 3 trajectory; (b) Case 3a - A segment of the trajectory.

It is notable that $S(x)$ is worst when x is very close to $\alpha \quad s.t. \quad x \to \alpha^+$. Therefore,

$$\sup_{x \in (\alpha, 2\pi - \alpha - \delta_\alpha]} C(x) = (1 + 2\pi - \alpha).$$

Case 3: Exit x is in the range $(2\pi - \alpha - \delta_\alpha, 2\pi]$ and is located by R_1. For the ease of calculations, we will further divide case into 3-a and 3-b (Figure 3.10).

We define y_α as the farthest location of exit for robot R_1 so that R_1 has time to catch R_2 while still on the circle i.e. a maximum time α. If δ_y is the chord travelled by R_1 to reach point A before R_2 turns on chord δ_α then δ_y is represented by the roots of

$$\delta_y = 2sin\left(\frac{\delta_y + 2y_\alpha}{2}\right).$$

Definition 3.2. *Let y_α be the maximum point on the disc where R_1 will be able to*

catch R_2 before turning in. Then y_α is the unique roots of

$$y_\alpha + \delta_y = \alpha.$$

Therefore, by replacing δ_y we get

$$y_\alpha + 2sin\left(\frac{\delta_y + 2y_\alpha}{2}\right) = \alpha.$$

Case 3-a: Exit $x \in (2\pi - y_\alpha, 2\pi]$. Since y_α is the optimum point away from 2π which will allow R_1 to catch R_2 at point A on the periphery before turning on the path AB then the value of y_α will be the largest when $y_\alpha + \delta_y = \alpha$ and point $A = cycle(\alpha)$. Therefore,

Search cost $S(x) = 2\pi - x$.

$\varepsilon(x) = \delta_{2\pi - x}$.

Therefore, $\boxed{C(x) = 1 + 2\pi - x + 2\delta_{2\pi - x}.}$

Case 3-b: Exit $x \in (2\pi - \alpha - \delta_\alpha, 2\pi - y_\alpha]$.

In this case, exit x is located between $(2\pi - \alpha - \delta_\alpha)$ and $(2\pi - y_\alpha)$ (refer to Figure 3.11). Robot R_1 locates the exit after R_2 has turned in on AB then, R_1 will catch R_2 somewhere on the line AB. Point A (Figure 3.11) is the location where R_2 turns

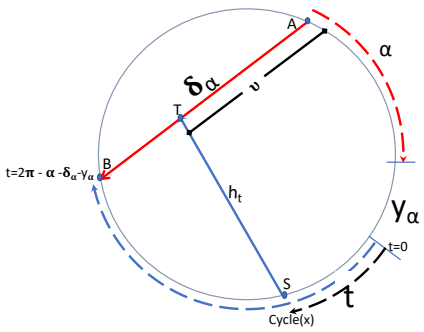

Figure 3.11: Case 3-b.

towards the line segment AB and

$$PointA = cycle(\alpha)$$

$$PointB = cycle(2\pi - \alpha - \delta_\alpha).$$

For this case, we re-parameterize the position of exit by introducing t such that $x \in (2\pi - \alpha - \delta_\alpha, 2\pi - y_\alpha)$. t is the location of exit from the last point on the circle where R_1 will catch R_2 on the periphery. Therefore location of exit,

$$x = 2\pi - y_\alpha - t$$

$$or \quad t = 2\pi - y_\alpha - x.$$

After catching R_2 somewhere on line AB, both R_2 and R_1will travel back to the exit for evacuation.

Therefore, value of t will vary as follows,

$$0 \leq t \leq 2\pi - y_\alpha - (2\pi - \alpha - \delta_\alpha)$$

$$or \quad 0 \leq t \leq \alpha + \delta_\alpha - y_\alpha.$$

Observation 3.3. Considering two distinct points $A = (a_1, a_2)$ and $B = (b_1, b_2)$ in $\mathbb{R}$, the trajectory of the robot moving along the line AB at a time t, can be given by the parametric equation

$$\text{line}(A, B, t) := \big(\frac{b_1 - a_1}{\|A - B\|}t + a_1, \frac{b_2 - a_2}{\|A - B\|}t + a_2\big).$$

A is the initial position of the robot, travelling from A to B with a unit speed.

If length of segment $T_\upsilon S$ is the distance travelled by the robot R_1 to catch R_2 then $T_\upsilon S$ can be given as $\|T_\upsilon - S\|$ where T_υ is a meeting point for R_1 and R_2 on line AB after time υ,

$$T_\upsilon = line\bigg(cycle(\alpha), cycle(2\pi - \alpha - \delta_\alpha), \upsilon\bigg).$$

Point S represents the exit located by robot R_1 so,

$$S = cycle(-y_\alpha - t).$$

Definition 3.4. *Let exit cycle(x) on the periphery, t such that $0 \leq t \leq \alpha + \delta_\alpha - y_\alpha$ then h_t defines the path travelled by R_1 to catch R_2. h_t, a function of t and α, is given as*

$$h_t = \|T_\upsilon - S\|.$$

At meeting point T_υ (Figure 3.11), R_1 and R_2 would take the same time therefore,

$$y_\alpha + t + h_t = \alpha + \upsilon$$

$$\upsilon = y_\alpha + t + h_t - \alpha$$

where $h_t = \|T_\upsilon - S\|$.

The placement of the exit x is a known value along with α. y_α and δ_α can be easily calculated since both are functions of α.

Computing h_t :

We notice that since $h_t = \|T_\upsilon - S\|$ so T_υ is a function of α and υ.

And $\upsilon = y_\alpha + t + h_t - \alpha$ so υ is a function of α and t.

It means that the non-linear equation $h_t = \|T_\upsilon - S\|$ needs to be solved to determine h_t. We find the roots through computation using Mathematica (see Appendix A).

Cost for locating the exit at $cycle(x)$, in range $[2\pi - \alpha - \delta_\alpha, 2\pi - y_\alpha]$,

Search cost $S(x) = y_\alpha + t$ (Figure 3.11)

$$= y_\alpha + 2\pi - y_\alpha - x$$

$$= 2\pi - x.$$

Evacuation cost $\varepsilon(x) = h_t$.

Total cost $C(x) = 1 + S(x) + 2\varepsilon(x)$. Therefore total cost,

$$\boxed{C(x) = 1 + 2\pi - x + 2h_t}$$

When we summarize all cases, cost

$$C(x) = \begin{cases} 1 + x + 2\delta_x, & 0 \leq x \leq \alpha \\ 1 + 2\pi - x, & \alpha < x \leq 2\pi - \alpha - \delta_\alpha \\ 1 + 2\pi - x + 2h_t, & 2\pi - \alpha - \delta_\alpha < x \leq 2\pi - y_\alpha \\ 1 + 2\pi - x + 2\delta_{2\pi - x}, & 2\pi - y_\alpha < x \leq 2\pi \end{cases} \tag{3.3}$$

3.5.3 Worst-case & Average-case Analysis

Above piecewise function $C(x)$ represents the total cost for many families of algorithms $\mathcal{B}_3$ with varying value of α. Since we do not have a closed formula and function needs to first calculate the roots therefore, we use Mathematica computation (for code see Appendix A) with the cost function (3.3) to numerically calculate the value of average-case cost, $Avg(C(x))$ for various values of α. First 2 cases of equation 3.3 are important from the worst-case perspective as they determine the worst-cost. From equation 3.1, worst-case cost can be calculated as

$$Wrs\big(C(x)\big) = \sup_{x \in [0, 2\pi)} C(x).$$

For all $\alpha \in [0, \pi]$, we have that

$$Wrs(\mathcal{B}_3) = \begin{cases} 1 + 2\pi - \alpha & , \alpha \leq \bar{\alpha} \\ C(\bar{\alpha}) & , \alpha > \bar{\alpha} \end{cases}$$

where $\bar{\alpha}$ is the solution to

$$1 + 2\pi - \alpha = C(\alpha_0).$$

So,

$$\begin{aligned} \bar{\alpha} &= 1 + 2\pi - C(\alpha_0) \\ &= 1 + 2 \cdot 3.14 - 5.74 \\ \bar{\alpha} &= 1.54 \end{aligned}$$

Here $\bar{\alpha}$ is the critical point corresponding to each α when the cost is worst, $\alpha \in (0, \pi)$. We know that α_0 represents the critical point corresponding to the worst-case cost in benchmark algorithm $\mathcal{B}_2$ (Figure 3.6).

$$Wrs(\mathcal{B}_3(\alpha)) = \begin{cases} C(\bar{\alpha}) & , C(\bar{\alpha}) > C(\alpha_0) \\ C(\alpha_0) & , C(\bar{\alpha}) \leq C(\alpha_0) \end{cases}$$

We discussed the cost of $cycle(\alpha)$ in detail but it is implicit that the cost of $cycle(\alpha)$ and $cycle(-\alpha)$ will be the same as they are mirror image on the axis and Case 3a

will be included in $cycle(-\alpha)$.

Average-case vs Worst-case values

From equation 3.2, average-case cost can be calculated as

$$Avg\big(C(x)\big) = \frac{1}{2\pi}\int_0^{2\pi} C(x)dx$$

Using Mathematica computation, we assembled the following table that shows the corresponding values of $Avg(\mathcal{B}_3)$ and $Wrs(\mathcal{B}_3)$ for various values of α

α	$Wrs(\mathcal{B}_3)$	$Avg(\mathcal{B}_3)$
0	7.2769	4.14149
0.5	6.78319	4.54365
1	6.28319	4.78305
1.5	5.78310	4.95007
2	5.73906	5.05315
2.5	5.73906	5.10357
3	5.73906	5.11704
π	5.73906	5.11728

Table 3: Varying values of $Avg(\mathcal{B}_3)$ and $Wrs(\mathcal{B}_3)$ for the corresponding value of α generated using Mathematica.

(a) Referring to figure 3.14, we notice that for each α as we go further in the distance cost is much less in the latter part of the curve which occurs due to the fact that both robots are moving towards each other so they are able to catch each other much quicker and evacuate as compared to the earlier part of the algorithm where it takes longer to locate and evacuate.

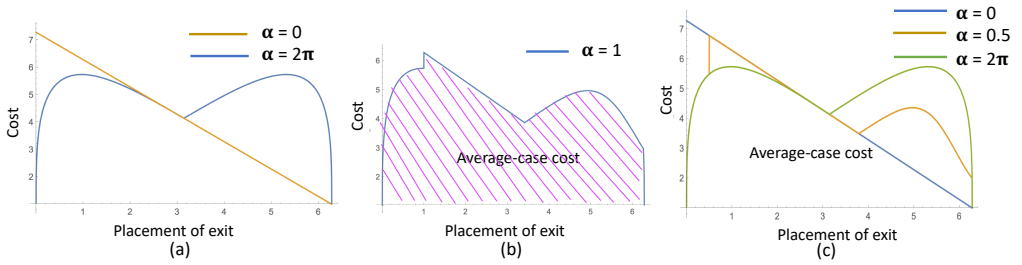

Figure 3.12: Various values of α for Algorithm $\mathcal{B}_3$: (a) plot for $\alpha = 0$ and $\alpha = 2\pi$ shows exactly the same graphs as $\mathcal{B}_1$ and $\mathcal{B}_2$ respectively; (b) plot for Algorithm $\mathcal{B}_3$ when $\alpha = 1$; (c) some more plots comparing the behaviour for Algorithm $\mathcal{B}_3$) when $\alpha = 0$, $\alpha = 0.5$ and $\alpha = 2\pi$.

(b) Function $C(x)$ follows the same algorithms same as benchmark algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$ for the extreme values of α, i.e. 0 and 2π, respectively. These plots numerically prove that our function $C(x)$ represents a spectrum of algorithms.

Figures 3.12(a) and 3.6(a) seem identical plots where 3.12(a) shows the behavior of algorithm $\mathcal{B}_3$ for the extreme values of α (0 and 2π) and 3.6(a) is a plot that compares the two previously defined benchmark algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$. Since all the other possible values of α must fall between 0 and 2π so we can say that $C(x)$ covers a wide spectrum of algorithms between 0 and 2π. Graph shows that the worst-case cost of $C(x)$ is between$\approx$ 7.2769 and $\approx$ 5.73906 as well as average-cost between $\approx$ 4.13945 and $\approx$ 5.11653. While figure 3.12(b) shows the complete plot for $\alpha = 1$, 3.12(c) represents multiple graphs for different values of α i.e. 0, 0.5, and 2π.

(c) Recall that our goal was to find an algorithm $\mathcal{B}_3(\alpha)$ among $\mathcal{A}$ where $\alpha \in (0, \pi]$

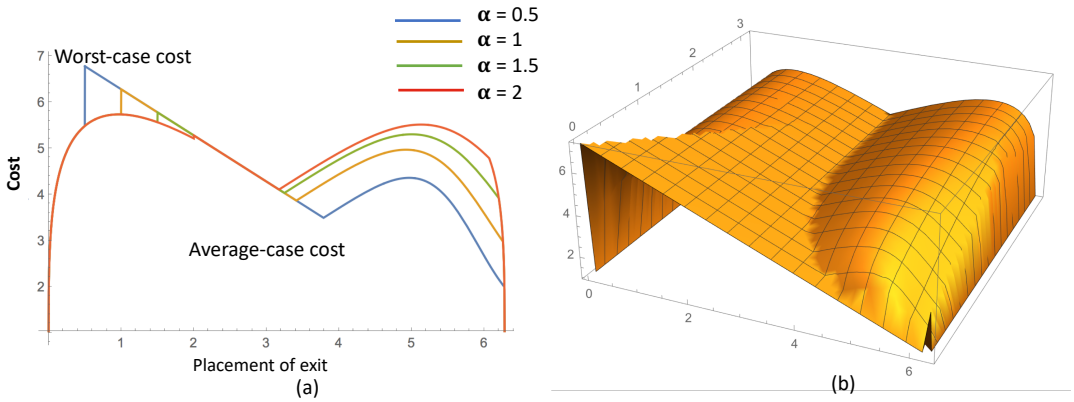

Figure 3.13: (a) Comparing the behaviour of Algorithm $\mathcal{B}_3$ for various values of α efficiency plot; (b) a 3-D plot showing the behaviour of Algorithm $\mathcal{B}_3$.

with $min\ Avg(\mathcal{A})$ such that

$$Wrs(\mathcal{A}) \leq \omega \ \ where \ \ \omega \in [5.74, 1 + 2\pi]$$

Referring to figure 3.14, $Avg(\mathcal{B}_3) \approx 4.95$. We further observe the behaviour of $C(x)$ in graph 3.13(a) that compares the plots for $\alpha = 0.5$, 1, 1.5 and 2. It is noticeable that the cost of algorithms remains largely similar to the $\mathcal{B}_2$ cost in the early phases, with one sharp increase towards the $\mathcal{B}_1$ cost. The jump is caused when the exit is placed just after the robot has turned inwards on the segment. The average-cost of the algorithms changes significantly in the latter phases of the algorithm as the placement of exit moves closer to 2π and the curve get closer to the $\mathcal{B}_1$ algorithm that has the lowest average-case cost. Figure 3.13(b) shows a 3-D plot representing the behaviour of Algorithm $\mathcal{B}_3$ for various values of α between 0 and 2π. In earlier section, we also defined *Efficiency* of the algorithm $\mathcal{A}$ as $\Big(Avg(\mathcal{A}), Wrs(\mathcal{A})\Big)$. Fol-

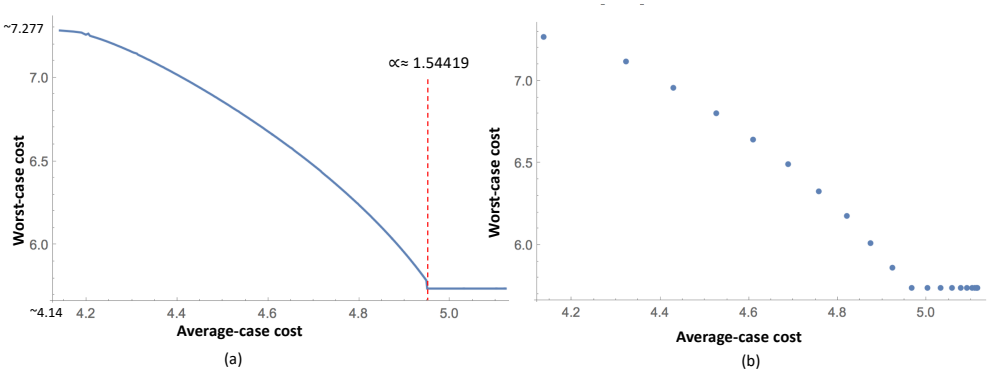

Figure 3.14: (a) Efficiency plot for various families of algorithms $\mathcal{B}_3$; (b) Point graph depicting the efficiency behaviour of Algorithm $\mathcal{B}_3$.

lowing the same concept of *Efficiency*, we plot $Avg(\mathcal{B}_3)$ and $Wrs(\mathcal{B}_3)$ for various families of algorithms. We obtain 3.14(b) point graph and when we compute more points closer to each other then we get 3.14(a) graph.

(d) Efficiency for the algorithm $\mathcal{B}_3$ starts to plateau when α is close to ≈ 1.54419. Referring to figure 3.14, we notice that when the worst-case value is low (≈ 5.7906) then average-case is least at (≈ 5.05315). At this point, worst-case cost is reached its max.

A minor variation in the value of α causes significant variation to the average-case cost closer to the worst-case cost. On the other hand, when the worst-case cost is low then any variation in worst-case causes minor variation in the average-case cost until $\bar{\alpha} = 1.54$. After that, worst-case cost remains unchanged even if there is an increase in the average-case cost.

Referring to figure 3.14, we notice that when the worst-case value is high (≈ 5)

Chapter 4

CONCLUSION

While the naive evacuation algorithm $\mathcal{B}_1$ sets a high goal (≈ 4.14) for the average-case cost in the evacuation problem with 2 robots on a unit disc but it's worst-case cost (≈ 7.276) is extremely high. Algorithm $\mathcal{B}_2$ manages to bring the worst-case cost down to (≈ 5.625) but the average-case cost increases to (≈ 5.1172). We notice the similar phenomenon of trade-off between the costs for cowpath problem, another famous problem in the field of search theory. We study a new evacuation algorithm $\mathcal{B}_3$ which represents a family of algorithms based on the varying values of α. We are able to devise a piecewise function to calculate the cost which we use to numerically calculate average-case and worst-case costs for the corresponding value of α using Mathematica when worst-case cost ≈ 5.73906. We conclude that $\mathcal{B}_3$ has the lowest possible average-case cost (≈ 5.05315) while keeping the worst-case cost (≈ 5.73906) to be low at $\alpha \approx 1.544$.

4.1 Future work

Although in a rather recent work by Chuangpishit et al. [26], new algorithms help reduce the average-case but problem is still open for an algorithm that has average value close to ≈ 4.14 while maintaining the worst-case cost between ≈ 7.276 and

≈ 5.625 for the 2-robots evacuation problem. Average-case/worst-case trade-offs for the randomized algorithms for evacuation problems needs to be studied. Other trade-off variations may include evacuation problems with:

- the consideration of the exit in an unknown search space
- multiple searcher robots
- multiple exits
- robots with speed other than unit

Appendix A

MATHEMATICA CODE FOR COMPUTING WORST-CASE AND AVERAGE-CASE COST FOR ALGORITHM $\mathcal{B}_3$

```
MYIntegrate(ff_, l_, u_) := Module[{temp=0, counter, prec=500., sizze},
    sizze= 1.(u−l)/prec;
    Do [
        temp = temp + ff [l + sizze * counter] * sizze,
        {counter, 0, prec}
        ];
        temp - sizze∗(ff(l)+ff(u))/2
       ]
  MYNMaximize(ff_, l_, u_) := Module[{temp=0, counter, prec = 500., sizze},
    sizze = 1.(u−l)/prec;
    Do [
        temp = max[temp, ff[l + sizze ∗ counter]],
        {counter, 0, prec}
       ];
        temp
        ]
```

cycle(angle_) := {cos(angle), sin(angle)};

line(a1_, b_, a2_, d_, t_) := {a1 . t + b, a2 . t + d};

linePoints(A_, B_, t_) := line$\left(\frac{B[[1]]-A[[1]]}{|A-B|}, \text{A[[1]]}, \frac{B[[2]]-A[[2]]}{|A-B|}, \text{A[[2]]}, \text{t}\right)$

δ(x_) := y/. FindRoot[y = 2sin($x + \frac{y}{2}$), {y, 1}]

y(a_) := yy/. Quiet[FindRoot[yy + δ(yy) = a, {yy,1}]]

TT(w_, a_) := linePoints(cycle(a), cycle(2 π - a - $\delta(a)$), w);

SS(t_,a_) := cycle(-yy(a)-t);

ww(x_,a_) := w/.

FindRoot[w = 2π - x + $\|\text{TT}(w, a) - \text{SS}(2\pi - \text{yy}(a) - x, a))\|$ -a, {w,1}]

hh(x_, a_) := ww(x, a) + a + x - 2π ;

simplealgo(x_) := 1 + x + 2$\delta(x)$;

costcase1(x_, a_) := 1 + x + 2$\delta(x)$;

costcase2(x_, a_) := 1+ 2π -x;

costcase3(x_, a_) := 1+ 2π -x +2 hh(x,a);

costcase4(x_, a_) := 1+ 2π - x + 2$\delta(2\pi - x)$;

cost(x_, a_) := Piecewise[{

{costcase1(x, a), $0 \leq x \leq a$},

{costcase2(x, a), $a < x \leq 2\pi - a - \delta(a)$},

{costcase3(x, a), $2\pi - a - \delta(a) < x \leq 2\pi - y(a)$},

{costcase4(x, a), $2\pi - y(a) < x \leq 2\pi$}

}]

NEWwrsa(a_) :=

Quiet[NMaximize[{cost(x, a), $0 \leq x \leq 2\pi$}, x][[1]]]

NEWavga(a_):=

Quiet [NIntegrate[{cost(x, a), {x, 0, 2π} }.$\frac{1}{2\pi}$]

Appendix B

REFERENCES

REFERENCES

[1] Alessandro Almeida, Geber Ramalho, Hugo Santana, Patrícia Tedesco, Talita Menezes, Vincent Corruble, and Yann Chevaleyre. Recent advances on multi-agent patrolling. In *Brazilian Symposium on Artificial Intelligence*, pages 474–483. Springer, 2004.

[2] S. Alpern. Hide and seek games. *Institut fur Hohere Studien, Wien*, 1976.

[3] S Alpern and M Asic. *Two search games on graphs*. London School of Economics and Political Science International Centre for Economics and Related Disciplines, 1982.

[4] S Alpern, R Fokkink, L Gasieniec, R Lindelauf, and VS Subrahmanian. Search theory, a game theoretic approach, 2013.

[5] Steve Alpern. Search games on trees with asymmetric travel times. *SIAM Journal on Control and Optimization*, 48(8):5547–5563, 2010.

[6] Steve Alpern and Shmuel Gal. Rendezvous search on the line with distinguishable players. *SIAM Journal on Control and Optimization*, 33(4):1270–1276, 1995.

[7] Steve Alpern and Shmuel Gal. The theory of search games and rendezvous. int. series in operations research and management science, 2002.

[8] Steve Alpern and Shmuel Gal. *The theory of search games and rendezvous*, volume 55. Springer Science & Business Media, 2006.

[9] Steve Alpern and Shmuel Gal. *The theory of search games and rendezvous*, volume 55. Springer Science & Business Media, 2006.

[10] EJ Anderson. Mazes: Search games on unknown networks. *Networks*, 11(4):393–397, 1981.

[11] Ricardo A Baezayates, Joseph C Culberson, and Gregory JE Rawlins. Searching in the plane. *Information and computation*, 106(2):234–252, 1993.

[12] Paul Balister, Amy Shaw, Bela Bollobas, and Bhargav Narayanan. Catching a fast robber on the grid. *Journal of Combinatorial Theory, Series A*, 152:341–352, 2017.

[13] Vic Baston and Shmuel Gal. Rendezvous on the line when the players' initial distance is given by an unknown probability distribution. *SIAM journal on control and optimization*, 36(6):1880–1889, 1998.

[14] Nadine Baumann and Martin Skutella. Earliest arrival flows with multiple sources. *Mathematics of Operations Research*, 34(2):499–512, 2009.

[15] Anatole Beck and DJ Newman. Yet more on the linear search problem. *Israel journal of mathematics*, 8(4):419–429, 1970.

[16] Bellman. Minimization problem. *Bull. Am. Math. Soc.*, pages 62–270, 1956.

[17] Richard Bellman. An optimal search. *Siam Review*, 5(3):274, 1963.

[18] Anthony Bonato. *The game of cops and robbers on graphs*. American Mathematical Soc., 2011.

[19] Anthony Bonato. *The game of cops and robbers on graphs*. American Mathematical Soc., 2011.

[20] Anthony Bonato, Xavier Perez-Gimenez, Pawel Pralat, and Benjamin Reiniger. The game of overprescribed cops and robbers played on graphs. *Graphs and Combinatorics*, 33(4):801–815, 2017.

[21] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. cambridge university press, 2005.

[22] Sebastian Brandt, Felix Laufenberg, Yuezhou Lv, David Stolz, and Roger Wattenhofer. Collaboration without communication: evacuating two robots from a disk. In *International Conference on Algorithms and Complexity*, pages 104–115. Springer, 2017.

[23] Jérémie Chalopin, Victor Chepoi, Nicolas Nisse, and Yann Vaxès. Cop and robber games when the robber can hide and ride. *SIAM Journal on Discrete Mathematics*, 25(1):333–359, 2011.

[24] Kamalika Chaudhuri, Fan Chung Graham, and Mohammad Shoaib Jamall. A network coloring game. In *International Workshop on Internet and Network Economics*, pages 522–530. Springer, 2008.

[25] Yann Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. In *Intelligent Agent Technology, 2004.(IAT 2004). Proceedings. IEEE/WIC/ACM International Conference on*, pages 302–308. IEEE, 2004.

[26] Huda Chuangpishit, Konstantinos Georgiou, and Preeti Sharma. Average case-worst case tradeoffs for evacuating 2 robots from the disk in the face-to-face model. *arXiv preprint arXiv:1807.08640*, 2018.

[27] Jurek Czyzowicz, Stefan Dobrev, Konstantinos Georgiou, Evangelos Kranakis, and Fraser MacQuarrie. Evacuating two robots from multiple unknown exits in a circle. *Theoretical Computer Science*, 2016.

[28] Jurek Czyzowicz, Leszek Gasieniec, Konstantinos Georgiou, Evangelos Kranakis, and Fraser MacQuarrie. The beachcombers' problem: walking and searching with mobile robots. *Theoretical Computer Science*, 608:201–218, 2015.

[29] Jurek Czyzowicz, Leszek Gasieniec, Thomas Gorry, Evangelos Kranakis, Russell Martin, and Dominik Pajak. Evacuating robots via unknown exit in a disk. In *International Symposium on Distributed Computing*, pages 122–136. Springer, 2014.

[30] Jurek Czyzowicz, Leszek Gasieniec, Adrian Kosowski, and Evangelos Kranakis. Boundary patrolling by mobile agents with distinct maximal speeds. In *European Symposium on Algorithms*, pages 701–712. Springer, 2011.

[31] Jurek Czyzowicz, Leszek Gasieniec, Adrian Kosowski, Evangelos Kranakis, Danny Krizanc, and Najmeh Taleb. When patrolmen become corrupted: Monitoring a graph using faulty mobile robots. *Algorithmica*, 79(3):925–940, 2017.

[32] Jurek Czyzowicz, Konstantinos Georgiou, Ryan Killick, Evangelos Kranakis, Danny Krizanc, Lata Narayanan, Jaroslav Opatrny, and Sunil Shende. God save the queen. *arXiv preprint arXiv:1804.06011*, 2018.

[33] Jurek Czyzowicz, Konstantinos Georgiou, Ryan Killick, Evangelos Kranakis, Danny Krizanc, Lata Narayanan, Jaroslav Opatrny, and Sunil Shende. Priority evacuation from a disk using mobile robots. In *International Colloquium on Structural Information and Communication Complexity*, pages 392–407. Springer, 2018.

[34] Jurek Czyzowicz, Konstantinos Georgiou, Evangelos Kranakis, Danny Krizanc, Lata Narayanan, Jaroslav Opatrny, and Sunil Shende. Search on a line by byzantine robots. *arXiv preprint arXiv:1611.08209*, 2016.

[35] Jurek Czyzowicz, Konstantinos Georgiou, Evangelos Kranakis, Fraser MacQuarrie, and Dominik Pajak. Fence patrolling with two-speed robots. In *ICORES*, pages 229–241. Citeseer, 2016.

[36] Jurek Czyzowicz, Konstantinos Georgiou, Evangelos Kranakis, Lata Narayanan, Jaroslav Opatrny, and Birgit Vogtenhuber. Evacuating robots from a disk using face-to-face communication. In *International Conference on Algorithms and Complexity*, pages 140–152. Springer, 2015.

[37] Jurek Czyzowicz, Kostantinos Georgiou, and Evangelos Kranakis. Group search and evacuation. In *Distributed Computing by Mobile Entities*, pages 335–370. Springer, 2019.

[38] Jurek Czyzowicz, Evangelos Kranakis, Danny Krizanc, Lata Narayanan, and Jaroslav Opatrny. Search on a line with faulty robots. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, pages 405–414. ACM, 2016.

[39] Pallab Dasgupta, PP Chakrabarti, and SC DeSarkar. A near optimal algorithm for the extended cow-path problem in the presence of relative errors. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 22–36. Springer, 1995.

[40] Erik D Demaine, Sándor P Fekete, and Shmuel Gal. Online searching with turn cost. *Theoretical Computer Science*, 361(2-3):342–355, 2006.

[41] Melvin Dresher. Games of strategy: theory and applications. Technical report, RAND CORP SANTA MONICA CA, 1961.

[42] Yehuda Elmaliach, Noa Agmon, and Gal A Kaminka. Multi-robot area patrol under frequency constraints. *Annals of Mathematics and Artificial Intelligence*, 57(3-4):293–320, 2009.

[43] Yehuda Elmaliach, Asaf Shiloni, and Gal A Kaminka. A realistic model of frequency-based multi-robot polyline patrolling. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1*, pages 63–70. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[44] Sándor Fekete, Chris Gray, and Alexander Kröller. Evacuation of rectilinear polygons. In *International Conference on Combinatorial Optimization and Applications*, pages 21–30. Springer, 2010.

[45] Shannon L Fitzpatrick and John Paul Larkin. The game of cops and robber on circulant graphs. *Discrete Applied Mathematics*, 225:64–73, 2017.

[46] Fedor V Fomin. Helicopter search problems, bandwidth and pathwidth. *Discrete Applied Mathematics*, 85(1):59–70, 1998.

[47] Peter Frankl. Cops and robbers in graphs with large girth and cayley graphs. *Discrete Applied Mathematics*, 17(3):301–305, 1987.

[48] S Gal and EJ Anderson. Search in a maze. *Probability in the Engineering and Informational Sciences*, 4(3):311–318, 1990.

[49] Shmuel Gal. Minimax solutions for linear search problems. *SIAM Journal on Applied Mathematics*, 27(1):17–30, 1974.

[50] Shmuel Gal. Search games with mobile and immobile hider. *SIAM Journal on Control and Optimization*, 17(1):99–122, 1979.

[51] Shmuel Gal. Search games with mobile and immobile hider. *SIAM Journal on Control and Optimization*, 17(1):99–122, 1979.

[52] Shmuel Gal. Computing elementary functions: A new approach for achieving high accuracy and good performance. In *Accurate Scientific Computations*, pages 1–16. Springer, 1986.

[53] Shmuel Gal. On the optimality of a simple strategy for searching graphs. *International Journal of Game Theory*, 29(4):533–542, 2001.

[54] Shmuel Gal. Searching a graph a working version. 2002.

[55] Shmuel Gal. Strategies for searching graphs. In *Graph Theory, Combinatorics and Algorithms*, pages 189–214. Springer, 2005.

[56] Konstantinos Georgiou, George Karakostas, and Evangelos Kranakis. Search-and-fetch with 2 robots on a disk: Wireless and face-to-face communication models. *arXiv preprint arXiv:1611.10208*, 2016.

[57] E. Gilbert. Games of identification and convergence. *SIAM Journal on Applied Mathematics*, 4(1):16–24, 1962.

[58] Noam Hazon and Gal A Kaminka. On redundancy, efficiency, and robustness in coverage for multiple robots. *Robotics and Autonomous Systems*, 56(12):1102–1114, 2008.

[59] Bruce Hoppe and Éva Tardos. Polynomial time algorithms for some evacuation problems. In *SODA*, volume 94, pages 433–441, 1994.

[60] Ming-Yang Kao, John H Reif, and Stephen R Tate. Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem. *Information and Computation*, 131(1):63–79, 1996.

[61] Michael Kearns, Siddharth Suri, and Nick Montfort. An experimental study of the coloring problem on human subject networks. *Science*, 313(5788):824–827, 2006.

[62] Athanasios Kehagias, Dieter Mitsche, and Paweł Prałat. Cops and invisible robbers: The cost of drunkenness. *Theoretical Computer Science*, 481:100–120, 2013.

[63] BO Koopman. Search and screening, operations evaluation group report 56. *Center for Naval Analysis, Alexandria, Virginia*, 1946.

[64] Steven Lalley and Herbert Robbins. Stochastic search in a convex region. *Probability Theory and Related Fields*, 77(1):99–116, 1988.

[65] Eric Langford. A continuous submarine versus submarine game. *Naval Research Logistics Quarterly*, 20(3):405–417, 1973.

[66] Aydano Machado, Geber Ramalho, Jean-Daniel Zucker, and Alexis Drogoul. Multi-agent patrolling: An empirical analysis of alternative architectures. In *International Workshop on Multi-Agent Systems and Agent-Based Simulation*, pages 155–170. Springer, 2002.

[67] Paweł Prałat and Nicholas Wormald. Meyniel's conjecture holds for random graphs. *Random Structures & Algorithms*, 48(2):396–421, 2016.

[68] Ryan A Rossi and Nesreen K Ahmed. Coloring large complex networks. *Social Network Analysis and Mining*, 4(1):228, 2014.

[69] E Anderson S Gal. he theory of search games and rendezvous interfaces. *International Journal of Game Theory*, 34(3):235–236, 2004.

[70] Vladimir Yanovski, Israel A Wagner, and Alfred M Bruckstein. A distributed ant algorithm for efficiently patrolling a network. *Algorithmica*, 37(3):165–186, 2003.