ASSIGNMENT AND TRAJECTORY PLANNING FOR TWO ROBOTS WITH SPORADIC COMMUNICATION AND RISK OF FAILURE

An Undergraduate Research Scholars Thesis

by

WILLIAM PARK

Submitted to the LAUNCH: Undergraduate Research office at
Texas A&M University
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by
Faculty Research Advisor: Dr. Dylan Shell

May 2023

Major: Computer Science

RESEARCH COMPLIANCE CERTIFICATION

Research activities involving the use of human subjects, vertebrate animals, and/or biohazards must be reviewed and approved by the appropriate Texas A&M University regulatory research committee (i.e., IRB, IACUC, IBC) before the activity can commence. This requirement applies to activities conducted at Texas A&M and to activities conducted at non-Texas A&M facilities or institutions. In both cases, students are responsible for working with the relevant Texas A&M research compliance program to ensure and document that all Texas A&M compliance obligations are met before the study begins.

I, William Park, certify that all research compliance requirements related to this Undergraduate Research Scholars thesis have been addressed with my Faculty Research Advisor prior to the collection of any data used in this final thesis submission.

This project did not require approval from the Texas A&M University Research Compliance & Biosafety office.

TABLE OF CONTENTS

Page

ABSTRACT

Assignment and Trajectory Planning for Two Robots with Sporadic Communication and Risk of Failure

William Park
Department of Computer Science and Engineering
Texas A&M University

Faculty Research Advisor: Dr. Dylan Shell
Department of Computer Science and Engineering
Texas A&M University

A common approach to distributing work to be completed simultaneously among different robots is called Multi-Robot Task Allocation. Particularly in a path planning scenario, two robots are assigned tasks to drive to two different destinations to perform some action, with emphasis on speediness. This paper tackles the real-world situation where some robots may fail, leading to any remaining robots reassigning their destinations to balance the differences in rewards and travel time between different locations. The catch is that when robots are restricted to sporadic communication due to environmental or resource constraints, surviving robots experience a delayed response to these failures – the uncertainty leading to many possible worlds where a late reassignment hinders making timely progress. We propose an approach where robots proactively plan their paths during these intervals of uncertainty in anticipation of possible robot failures. Specifically, robots will answer the question: to what location should I drive before my team's next time of communication to best react on average to possible news of their failures? Observe the case when one robot will definitely fail – the other must drive directly to a particular destination. But when that same robot will definitely live, the other might drive to a different destination to balance the team-wise reward.

For the case when that robot's probability of failure is uncertain, we present that the surviving robot's trajectory will incorporate a path that lies somewhere in between the prior two. We give examples where this is the best plan that handles delayed awareness of robot failure because, over multiple trials, it will ultimately travel less while collecting more reward.

DEDICATION

To my parents who supported me through my college career. It is their love that had my back when even my own back got injured.

ACKNOWLEDGMENTS

Contributors

I would like to thank my faculty advisor, Dr. Shell, for his guidance and support over the past four semesters. The time, advice, and encouragement he has given me are immensely appreciated, and I enjoyed learning about writing techniques and robotics from him.

Thanks also go to my friends and colleagues and the department faculty and staff for making my time at Texas A&M University a great experience.

Finally, thanks to my family for their encouragement throughout my education.

Funding Sources

Undergraduate research did not receive funding.

1. INTRODUCTION

Things in life often don't go the way we plan them to. Even worse — we may find out too late that we deviated from our plans a long time ago!

Often, a late change in plans causes us to expend more time and energy than having known about the change beforehand. Making progress towards plan A and changing to plan B expends more energy than doing plan B from the beginning. In this paper, we explore the consequences of delayed changes to plans in a Multi-Robot Task Allocation problem — a class of problems where robots are each assigned tasks from a collection of tasks in the most efficient way possible.

Specifically, we would like to know, if robots have statistics about the future beforehand — is there a different method of approaching their tasks that will allow them to anticipate changes in the future? The problem becomes more clear in our particular domain of trajectory planning — an instance of the Multi-Robot Task Allocation problem where mobile robots are assigned to drive to different cities and do some actions to collect some reward. Changes to an assigned plan arise due to some subset of robots failing before arrival. If robots have access to real-time communication, they would be able to readjust their paths based on these changes, but in our scenario, communication must be sporadic instead. This induces intervals of time where all living robots are uncertain about the state of the rest of their teams before they communicate to check in. Here, the question proposed becomes evident — if robots have statistics on the likelihood of all robot failures, then is there a different method of driving to their assigned destination that will put them in a better position to react to changes, i.e. switch destinations?

The problem constraints that are to be studied are problems that crop up often in the real world. As advanced as robotics has gotten these days, it is still expected of robots to experience failure at some point. This is because the environment robots are embodied in is often times complex and unforgiving and even the systems that compose the robots themselves might be subject to faults. For a team of robots, it is a worldly expectation that at least one will fail. It is then no

surprise that finding approaches that coexist with some robot failures is desired since we would not want the entire team of robots to collapse just because one of them malfunctioned. For Multi-Robot Task Allocation, the adaptation process to robot failure is straightforward: recompute the assignment of remaining robots to tasks and have the robots go on with their updated tasks.

The hard case is the sporadic communication constraint. Issues like these naturally arise due to environmental or resource constraints where real-time communication is not possible or viable. For example, consider the underwater navigation scenario where two aquatic, mobile robots are tasked with swimming to and investigating two coral reefs of differing informative value. Because water isn't an ideal medium for communicating and robots are sufficiently far apart where this becomes a big problem, we want robots to schedule in advance times to surface above the water in order to communicate while accessing terrestrial devices such as satellite communication [1]. We also want to incorporate this check-in within robots' plans because they might be swimming in a region of water where turbulent currents pose a risk of sweeping the robots away or causing malfunctions so checking in on each other is a necessary part of making sure they adapt to get the most informative value possible [2].

A resource constraint example might be robots deliberately restricting the number of times they communicate to a finite set of points in time in order to preserve energy like their battery. This example is highly applicable to many scenarios where some failure is possible and conserving energy is a potent issue. A key clarification to be made is that sporadic communication is intentionally scheduled rather than experienced randomly and haphazardly. Unlike the random, unknown times a phone in a tunnel will connect and disconnect from Wifi, robots have the precise time they will communicate because they plan them ahead of time. The focus of our problem is only on forecasting what event will occur at a specific future time.

More generally, a realistic problem encountered in real life is Multi-Robot Task Allocation when communication between robots is unreliable and environmental changes occur that affect future rewards. We explore planning for the communication model where robots are disconnected for a while and later achieve global communication to dispense any unsent changes observed in

the environment. Some connectivity is important because it enables task-allocation algorithms to realign robots based on the communicated changes toward a new optimal objective. The question we seek to answer is if robots can forecast the probability of those changes occurring, what actions should they take while not connected so that when they eventually relay any changes, the robots realign and maximize their reward on average? We investigate this question for robots allocated to destinations in a 2-D plane where a subset of robots failing is expected.

The common research theme underlying this question is the belief that doing additional work ahead of time helps robots react to uncertain events in the future since we trust being prepared for the unexpected is better than not. Currently, there are not many scholarships related to Multi-Robot Task Allocation addressing these specific conditions — sporadic communication and robot failure — proactively. The research will contribute a proactive strategy to this field for the trajectory planning instance of the problem.

1.1 Related Works

Turpin et al. [3] demonstrate in the deterministic 2-D setting that utilizing task allocation can help assign trajectories for n homogeneous robots to n destinations that minimize the total distance traveled. Their method uses the Hungarian Algorithm to find the trajectories whose distances sum to be the shorted combined distance possible. Our research also follows this trajectory generation idea where we compare optimal trajectory allocations for each subset of robots due to robot failure and consider paths that maximize the expected reward across the subsets.

Liu and Shell [4] give a Hungarian algorithm variant that hands each robot an interval that thresholds how much deviation from their current assignment they are willing to take before issuing a global reallocation. The method in [4] leaves the magnitude of the violation for the reallocation to handle; our approach seeks to minimize the variation of an anticipated violation.

A successor work to [4], Nam and Shell [5] approach the problem of uncertain rewards and expensive communication that could make connectivity prohibitive with the philosophy that doing more upfront work leads to better performance at run-time. They propose an upfront sensitivity analysis to scope out cost regions for when current allocations become suboptimal and suggest

corrective methods that take advantage of it during run-time. These corrective methods are: dividing robots into sub-teams that only need to communicate internally, persisting with the current allocation if it outweighs the cost of communicating, and communicating incrementally. These methods allow the team to react to robot failures by reallocating with minimal communication or knowing to persist if communicating is prohibitive. The difference between our research is that in [5] they don't make forecasts about changes that could happen and deal with that by optimizing — during planning time — the robots' ability to react. In our research, we do assume these and focus specifically on where robots should proactively move during run-time to best discover changes, a communication event that could be prohibitive to the problem in [5]. Our research is similar because our method of proactive movement to be better positioned for future changes takes inspiration from doing sensitivity analysis ahead of time to have advanced knowledge of how to react.

In the field of pursuit-evasion, Olsen et al. [6] study the problem where pursuer robots are tasked with detecting an evader within a two-dimensional polygonal environment. The authors propose a novel formulation of the pursuit-evasion problem to tackle the case when at least one robot fails. The plan incorporates redundancy amongst the paths robots take ensuring the search is not disturbed by any particular robot failure. The key similarity with our research is that we both incorporate an idea of redundancy into our robots' paths. Their redundancy is geometrical — no matter which robot fails, the pursuers are guaranteed to search the entire space for evaders — whereas our redundancy is probabilistic — based on forecasted data we ensure the robot's path will get us the most reward. One key difference is that our robots need to communicate in order to adapt, whereas, their robots do not communicate and will achieve success by design.

An alternative approach for partially observable and decentralized conditions is multi-task multi-agent deep reinforcement learning [7]. Here robots each learn a planning policy from experience by training agents to coordinate without communication, although robot failure is not considered.

2. METHODS

This section provides a mathematical formulation of the Multi-Robot Task Allocation Problem for n robots under sporadic communication constraints and robot failures. The analysis is specified for two robots and derives and computes a trajectory that optimizes the expectation of rewards robots collect as well as a description of a plan for the team.

2.1 Problem Formulation

2.1.1 Multi-Robot Task Allocation

Robots in the set of n robots, $R = \{r_1, \ldots, r_n\}$, are each designated to move to a different destination from a set of locations, $D = \{d_1, \ldots, d_n\}$, so that there is a one-to-one correspondence. Their task isn't considered finished until they have arrived at their destination and they do some 'action' to signify completion. The latter condition is supposed to allow cases when a robot arrives at its destination but might want to wait for any unforeseen circumstances before deciding to move elsewhere.

Task completion is followed by a reward whose quantity corresponds to the location it was completed at, denoted by the map $M : D \to \mathbb{R}$. Then it is added to the negative reward incurred from the time it took for that robot to move there. The linear function of that time to a real number, $\phi : [0, \infty) \to \mathbb{R}$, is the fee paid for travel expenses. The total reward the robot gets for completing its mission is:

$$\psi(d, t) = M(d) - \phi(t). \tag{1}$$

One example of a reward is informative value. In the coral reef example from the introduction, different coral reefs may have different informative values, i.e. coral reefs rarely visited have more potential for new information than coral reefs visited often. The time it takes to travel to a coral reef can be translated as the penalty for expending resources to gather information — imposing a real-world trade-off between values and constraints. The example demonstrates the applicability of the general problem to a broad domain.

Although the goal of the problem is to maximize the collective sum of rewards from the team of robots, the catch is that at some moment along the way a subset of robots can fail — eliminating them and their ability to finish their task for good. If the other robots continue none the wiser, they risk the unfortunate circumstance of being sub-optimal.

Because robots fail randomly, we must consider every possible way robots can fail. The ultimate goal then is to maximize the expected collective reward acquired from handling each event where a different subset of robots fail.

2.1.2 Communication Model

Fortunately, the team is allotted a collection of k times from the set $T = \{\tau_1, \ldots, \tau_k\}$ for a chance to contact everyone to see who's still alive and update their assignment accordingly. Because there is a limit in the time window of when failure is possible, we assume that robots will schedule their last time to communicate, τ_k, to occur after that.

2.1.3 Failure Model

In conjunction with delayed contacts, anytime during the interval $[0, \tau_k]$ a robot has a chance of failing. Specifically, the probability density function of this phenomenon is modeled by an **exponential distribution** — where the event of failure occurs independently and continuously over all time. Additionally, the model captures the nature that robots can only fail once. We believe this is a good approximation of the failure process because although the reasons for failure at any point in time are various, the number of occurrences over an interval should follow a certain trend which can be captured by the *rate parameter* of the distribution.

At communication time τ_i, robots obtain the most recent information about the state of failures amongst the team. Anticipating this information could change in the future, robots can leverage the failure model to forecast the probability that a robot will fail at their next scheduled communication time τ_{i+1}, to inform themselves what the best move should be in the time prior. We can model the forecasting as a weighted coin flip for each robot at time τ_{i+1} since there are only two possibilities possible for it. If the coin flips heads, the robot lives, otherwise it fails. The

weight to flip tails for a robot, p_{tails}, is the integral of the exponential distribution from τ_i to τ_{i+1}. The cumulative distribution function, with a rate of failure parameter λ,

$$F(x;\lambda) = \begin{cases} 1 - e^{-\lambda x} & x \geq 0, \\ 0 & x < 0 \end{cases} \tag{2}$$

can be used to derive the following probability of flipping tails

$$p_{\text{tails}} = F(\tau_{i+1};\lambda) - F(\tau_i;\lambda). \tag{3}$$

In the instance of K communication checkpoints, the chance of failure is conditioned on the robot's previous survival. If T is the exponential random variable that is the time of robot failure, and $\Delta t = \tau_{i+1} - \tau_i$, then the probability of failing in $(\tau_i, \tau_{i+1}]$ conditioned on prior survival, $\Pr(T > \tau_i)$, is

$$\Pr(T \leq \tau_{i+1} \mid T > \tau_i) = \Pr(T \leq \Delta t) \tag{4}$$

by memorylessness. Memorylessness makes the coin-flipping model work out when you weight each coin based on the integral over the duration of time passed for each interval because the duration implicitly conditions the weight on prior survival. For example, if a robot will communicate two times, the weight of failure at the second communication is the probability of flipping heads on the first communication, multiplied by the probability of flipping tails on the second.

Note that we make two seemingly contradictory statements between the communication model and the failure model; the communication model says failure will stop being possible after time τ_k but the failure model says that the failure possibility extends to infinity in equation 2. To clarify, the failure density is a statement about how the hazard would behave if it extended to infinity. But we assume a limit to the window of failure, so we can cut short the possibility of failure at some time τ_k. Therefore, there is no contradiction.

2.1.4 Robot State Model

The robot can be in three states at any time t: alive, dead, or finished. The true state of each robot is marked by $G : [0, \infty) \to \{\text{ALIVE}, \text{DEAD}, \text{FINISHED}\}^n$. All the robots start alive — $G(0) = \{\text{ALIVE}\}^n$ — and once a robot dies it stays dead — for the first time of death $t_d^{(i)} = \inf\{t \mid G(t)_i = \text{DEAD}\}$, $G(t)_i = G(\min(t, t_d^{(i)}))_i$. Robots die according to the exponential, hazard function in the failure model. Robots are finished when they complete their task before they die.

The problem is that robots do not know the true state G, so they must approximate it with their subjective, delayed knowledge A, where $A : [0, \infty) \to \{\text{ALIVE}, \text{DEAD}, \text{FINISHED}\}^n$. Because robots are distributed, knowledge of who's alive may only be acquired and disseminated during the allotted K checkpoints, denoted for time τ_i as $A(\tau_i) = G(\tau_i)$. Only until the next call can they update their knowledge. That knowledge in the interval between two calls at times τ_i and τ_{i+1}, where $t_{\text{interval}} \in [\tau_i, \tau_{i+1})$, is characterized as $A(t_{\text{interval}}) = A(\tau_i)$. Accordingly, robots are all in sync at the start: where $A(0) = G(0)$.

2.1.5 Trajectory Model

The trajectory a robot follows is a function from time and knowledge, to a location in the plane — $\pi_i : [0, \infty) \times \{\text{ALIVE}, \text{DEAD}, \text{FINISHED}\}^n \to \mathbb{R}^2$ — where index i corresponds to the traveling robot. Denote the position as $\pi_i(A(t)_i)$.

To get a function that tells the running-cost a robot accrues at any time t, make a function C_i, where $C_i : [0, \infty) \to \mathbb{R}$, that considers the time a robot finishes: $t_f^{(i)} = \inf\{t \mid G(t)_i = \text{FINISHED}\}$. Then $C_i(t) = \phi(\min(t, t_f^{(i)}))$.

2.1.6 General Objective Function

Then the general utility function over the set of finished robots, $F = \{i \mid G(T)_i = \text{FINISHED}\}$, is

$$U = \sum_{i \in F} M(\pi_i(A(T)_i)) - C_i(T). \tag{5}$$

The objective function we wish to maximize is the expectation of the utility —

$$\mathbb{E}[U] \tag{6}$$

— over the probability distribution of the failure model. Specifically, we wish to know what the optimal policy, π_i^*, should be to maximize the expected utility.

We count only the finished robots' rewards because the goal of the problem is to maximize the sum of rewards of all alive robots that complete their mission. Any expenses accrued from dead robots are disregarded since we don't count something out of a robot's control — failure — against them. For example, if a robot explodes halfway towards its goal, it doesn't matter how much fuel has been spent prior — the whole fuel tank's gone! And the robot's trajectory policy isn't to be blamed for that.

In the same vein, if a robot happens to complete a task but fails before it is able to notify the rest of the team, the reward cannot be counted since there is no way for some central authority or the remaining robots to know about the success. This affects the remaining robots because one robot, assuming radio silence meant that the failed robot didn't finish the task, could decide to switch to the supposed incompleted task left behind, only to discover when it arrives that the task had been completed. We let this discovery count as a task completion in and of itself, to simplify the robot's behavior from having to switch again. Also, it makes sense since discovering the team obtained a reward is a reward in itself.

Additionally, it is important to emphasize that time is the factor being balanced against money, rather than distance. The reason is that the timed nature of incoming information from communication becomes irrelevant when all the robot has to do is wait until the final communication before departing with the full knowledge. Because there is no penalty to waiting since all that matters is the distance traversed, a strategy like this negates the thesis' goal of understanding how uncertainty over time impacts the way robots should move.

Assume from here on out that the paper will discuss the $n = 2$ case only. The explicit

formula for the expected utility is not written for the general case but is written in the case with two robots and a single checkpoint in section 2.2.

2.1.7 Optimal Trajectory

There are two properties of an optimal trajectory we must show: it is composed of straight line segments and the points connecting two line segments have restrictions to where they can be.

Suppose a robot communicates at τ_i and moves to another location by the time τ_{i+1}. This is simply going from one point to another, and the simplest and fastest way to represent the path of that motion is a straight line between them (supplemented with waiting if the robot arrives early). It is prudent to represent it this way because we want the robots to travel as fast as they can to a destination at τ_k. Additionally, we demonstrate next that there is no alternative to a straight path for trajectories between checkpoint times either.

Given the robot is positioned on some point, P, at checkpoint time τ_i, the locations that the robot can be at time τ_{i+1} must be within the area of the circle of reachable points centered around P. Given the robot moves at some velocity v_i, the radius of this circle must be $\frac{\tau_{i+1}-\tau_i}{v_i}$ units of distance. For simplicity, we assume all $v_i = 1$.

We consider a robot moving to a point — point P — that resides strictly in the circle interior as waiting because there is nothing else to do after arriving at point P until the next checkpoint. This implies no matter what coordinate point the robot moves to, the running-cost at the next checkpoint is the same — the fee $\phi(\tau_{i+1} - \tau_i)$. Then we can ask where an optimal P should be located during this window.

Lemma 2.1.1. *Specifically, the optimal location must reside on the arc of the circle to reach either destination with the shortest distance. This holds when the circle does not intersect the line* $\overleftrightarrow{d_1 d_2}$.

Proof. The proof is by contradiction - given a point Q inside the circle, we can always show an alternative point Q' on the circumference of the circle that is better than it. By taking the projection of Q onto the circumference to form Q', moving down the line passing through Q that is perpendicular to the line connecting the two destinations, you can use the Pythagorean theorem

to show that the distance from the projected point Q' to either city is always less than or equal to the corresponding distances to each city from Q, the point inside the circle.

Referencing Figure 1, let height $h = ||Q - A||$ and $h' = ||Q' - A||$. We know $h' < h$ by the definition of orthogonal projection. Then their squares must be less than each other:

$$(h')^2 < h^2.$$

Adding the base length X of triangle ΔQAd_1 to both sides,

$$(h')^2 + X^2 < h^2 + X^2.$$

By the Pythagorean theorem, we can rewrite the inequality in terms of the hypotenuses as

$$(a')^2 < a^2.$$

This implies $a' < a$ proving that the projection point Q' is closer to d_1 than Q. We can reapply the same argument to b' to get the same result. Notice that the argument makes no statement about the type of triangle so the proof generalizes to any triangle types not found in Figure 1. □

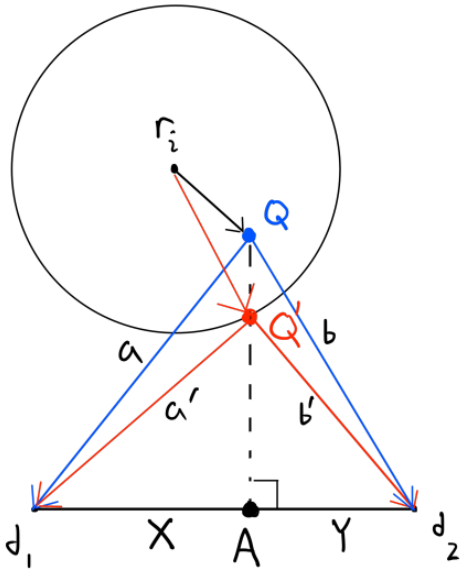

Figure 1: Shows the orthogonal projection Q' from Q towards the line segment connecting the destinations. Distances a and b are how far it would take to move to destination d_1 and d_2 from Q. Distances a' and b' are how far it would take to move to destination d_1 and d_2 from Q'. X is the length of the base formed by triangles ΔQAd_1 and $\Delta Q'Ad_1$. Y is the length of the base formed by triangles ΔQAd_2 and $\Delta Q'Ad_2$. The Pythagorean theorem T $a' < a$ and $b' < b$.

The intuitive meaning is that there is no reason to stop and wait prematurely for the next checkpoint time. A robot should always use the full time between τ_i and $\tau_{i=1}$ to get as far away from its starting point P and as close to either city as much as possible - the circumference being the outermost possible region for the robot to move to.

This argument generalizes to the multiple communication case.

Lemma 2.1.2. *Given an arbitrary path taken, you can improve upon it by reapplying the projection method from Lemma 2.1.1 which not only improves on the point you're projecting down but also the points on the subsequent path. This holds when the no τ-circles intersect the line $\overleftrightarrow{d_1 d_2}$.*

Proof. This is possible because every projection line is parallel so all points on the path will get improved simultaneously. Then by repeating the process for later points, you end up with a path

that goes to every circumference possible. In Figure 2 an arbitrary path is shown that utilizes waiting at a point before the next communication. The subsequent figures, Figure 3 and Figure 4 show the shifting process in order. This proves moving to the circumference without utilizing waiting will garner more reward than a strategy utilizing waiting, no matter how many communication checkpoints there are. □

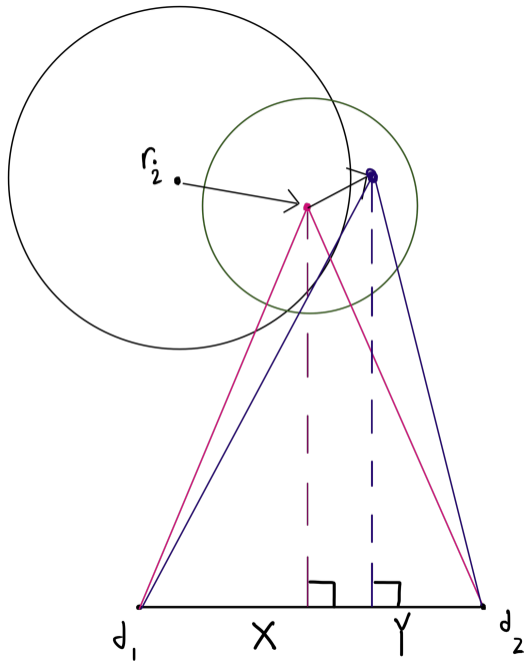

Figure 2: Shows the path a robot takes denoted by the black arrows. Each circle around a point represents the circle of reachable points until the next communication. Here, the robot moved to the pink point, waited until the next communication, and moved to the blue point. The triangles can be interpreted similarly to Figure 1 as the distances to either destination. Their heights are both orthogonal to the same base $\overline{d_1 d_2}$.

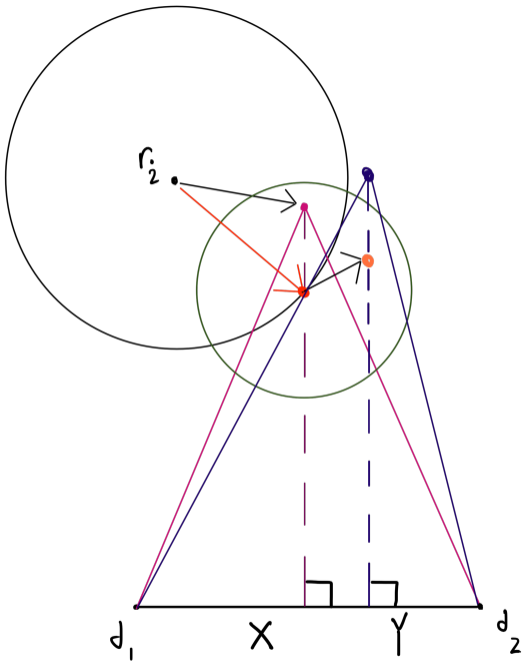

Figure 3: Follows from Figure 2 where the motion starting from the pink point is shifted perpendicularly downwards to the circumference of the first, black τ-circle. The second, green τ-circle and the motion from the pink point are translated downwards but their transformation stays the same. Notice the new red and orange points which represent potential points of departure towards either destination. Because they lie on the same perpendicular height-lines as before, we can apply Lemma 2.1.1 to show any departure travels less distance than without the shift. The red arrow shows the improved path the robot should take to get to the red point over the previous path.

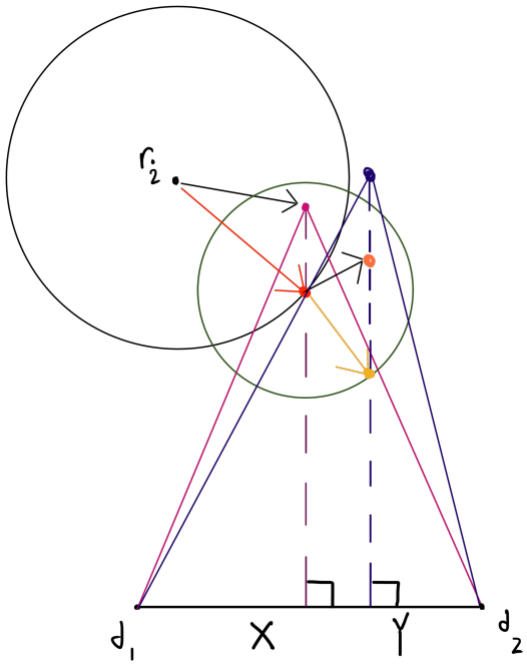

Figure 4: Follows from Figure 3 which demonstrates the iterative process of improving on a subsequent τ-circle — the green circle here — following the logic of Lemma 2.1.1. The red and yellow arrows show the new path that is proven to have departures (at the arrow tips) closer in distance to either destination than the previous path.

Because the points must always reside on the circumference, it is not possible to reach the circumference in time $\tau_{i+1} - \tau_i$ unless the robot goes in a straight path to the point on it. An exception to this argument is when the circle of reachable points intersects with the line segment between two destinations since moving past the line travels an unnecessary, extra distance.

The goal of specifying these two properties of an optimal trajectory is to simplify the problem to the question of where should these points connecting line segments be placed?

2.1.8 Optimal Assignment

Lemma 2.1.3. *There is always one robot that never switches its assignment regardless of if the other fails or not.*

Proof. To prove by contradiction assume any arbitrary starting points for the two robots at the final

checkpoint time τ_k. Assume that they are in the case when both robots live. They must choose one possible assignment out of the other. Because an optimal assignment is chosen, that means the reward from one assignment must be greater than the other, that is

$$\psi(d_1, t_{f_{11}}) + \psi(d_2, t_{f_{22}}) > \psi(d_1, t_{f_{21}}) + \psi(d_2, t_{f_{12}}) \tag{7}$$

where $t_{f_{ij}}$ denotes the time spent traversing to the destination by the $i-$th robot to destination d_j.

Now imagine the case when either of the robots fail. If r_2 failing leads r_1 to move to d_2, r_1 failing leads r_2 to move to d_1, and we assume these are the optimal decisions, the following must be true about the rewards:

$$\psi(d_2, t_{f_{12}}) > \psi(d_1, t_{f_{11}}) \tag{8}$$

and

$$\psi(d_1, t_{f_{21}}) > \psi(d_2, t_{f_{22}}). \tag{9}$$

Adding equations 8 and 9 you get

$$\psi(d_1, t_{f_{21}}) + \psi(d_2, t_{f_{12}}) > \psi(d_1, t_{f_{11}}) + \psi(d_2, t_{f_{22}}) \tag{10}$$

which is a contradiction to equation 7 which we decided was an optimal assignment but equation 10 would suggest otherwise. Therefore, our assumption that all robots must switch must be false, and thus, at least one robot never switches. We refer to the robot that could switch or not as the "Choice Bot" as its decision depends on the optimal strategy. Finally, a caveat is that I can only prove this for the single communication case. □

There are two immediate consequences of lemma 2.1.3. First is that there must be 4 different strategies robots utilize - there are 2 different ways to assign two alive robots and 2 ways to assign the Choice Bot when only it is alive, making a permutation of 4 total. Second is that if a robot never switches, then the optimal trajectory for it must be a straight path from start to finish.

2.2 Problem Definition for Two Robot, Single Communication Case

2.2.1 Objective Function

In the 2 robots, 2 destinations, and one communication time case, there are four objective functions depending on the assignment, assuming p_1 and p_2 are the probabilities the coin flips heads for each robot and the destination that at least one robot never switches from is d_2 (assume $M(d_2) \geq M(d_1)$):

"Never-Switch" — r_1 always goes to d_1 and r_2 always goes to d_2:

$$p_1\,\psi(d_1, ||d_1 - r_1||) + p_2\,\psi(d_2, ||d_2 - r_2||). \tag{11}$$

"Switch-If-Needed" — r_1 goes to d_1 if r_2 lives. r_1 goes to d_2 otherwise. r_2 always goes to d_2:

$$p_1\,(p_2\,\psi(d_1, ||d_1 - q_1||) + (1 - p_2)\,\psi(d_2, ||d_2 - q_1||) - \tau) + p_2\,\psi(d_2, ||d_2 - r_2||). \tag{12}$$

"Never-Switch Swapped" — r_1 always goes to d_2 and r_2 always goes to d_1:

$$p_1\,\psi(d_2, ||d_2 - r_1||) + p_2\,\psi(d_1, ||d_1 - r_2||). \tag{13}$$

"Switch-If-Needed Swapped" — r_2 goes to d_1 if r_1 lives. r_2 goes to d_2 otherwise. r_1 always goes to d_2:

$$p_2\,(p_1\,\psi(d_1, ||d_1 - q_2||) + (1 - p_1)\,\psi(d_2, ||d_2 - q_2||) - \tau) + p_1\,\psi(d_2, ||d_2 - r_1||). \tag{14}$$

It is already assumed that the optimal points for robots to move to for equations 11 and 13 lie on the straight line from start to destination because, if robots never switch their initial assignment, then the straight path must be the optimal one. For equations 12 and 14 we will use calculus methods to find the optimal locations, q_1 and q_2, to move to for each corresponding strategy. Finally, we will take the max over all objective functions to find the optimal strategy and

its corresponding optimal location.

2.2.2 Reduction to Polar Coordinates

Let's focus on the general form of objective functions 12 and 14. When maximizing these functions with respect to some point q, we can get rid of constant terms and scalar factors that never change. This boils the form of each equation down to:

$$-p\,||d_1 - q|| + -(1-p)\,||d_2 - q|| \tag{15}$$

where q is restricted to some point along the circle. Then the maximization problem of this objective function becomes a minimization problem when you take out the negative.

$$\begin{aligned}
&\underset{\theta}{\text{minimize}}\; p\,||d_1 - q|| + (1-p)\,||d_2 - q|| \\
&\text{subject to} \\
&||q - r|| = \tau
\end{aligned}$$

This is important because we can convert optimizing for a point in two dimensions into a problem of one dimension with polar coordinates where the polar coordinate of q can be denoted by (τ, θ) assuming we center the coordinate frame with respect to the robot in question.

After we simplified the equation we used numerical methods to solve the optimization problem due to being unable to find an algebraic expression for the solution. The attempt at solving the problem analytically is in the appendix below.

2.2.3 Numerical Methods

We used calculus methods to solve the objective equations: for each objective, we took the derivative to find the critical points, found the critical points at the greatest local maxima, and took the maximum over all objective function maxima to select the best strategy. We used a software package to implement the solution process.

2.3 Rationalizing under Uncertainty

This problem is about what would be the rational actions for robots to take under uncertainty — an open-ended question with many models and at many times, the choice between models is subject to a human-value judgment.

In our model, for rationality, our robots follow the principle of **maximum expected utility** — where in our particular instance the utility is dollars earned, and agents seek to maximize the average returns over all possible states of team-wide robot failure. This model is generally regarded as the starting framework for what rational agents should do when faced with uncertainty. A justification for using this framework is that you can derive it from a set of principles we regard as rational: orderability, transitivity, continuity, substitutability, monotonicity, decomposability [8].

Another problem in our decision-making we made was to make the reward of dollars earned **risk-neutral**. This means the utility value of money is its face value — one-to-one linear. This is important to highlight because there are other ways of modeling the utility of money such as taking the logarithm of money as the utility function to represent diminishing returns [8]. This formulation characterizes agents who are **risk-averse** and the choice is backed up by empirical studies. There is also another curve for **risk-seeking** agents. Our choice to stick with being risk-neutral was made because we didn't have a reason for robots to prefer being risky or not as that detail was not pertinent to our main goal of answering what location should robots move to, to anticipate uncertainty.

As an aside, the **optimizer's curse** is a phenomenon where the optimized solution based on estimated values can be sub-optimal when applied in the real-world [8]. If we make expected utilities from estimates rather than their true values, selecting the choice that leads to the highest expected utility can result in an overestimation of the true value. While this phenomenon can occur in some optimization processes, it does not apply to us since we use the exact expected utility to compare our choices, rather than estimated values.

Alternative models for rationality may be to minimize regret or to take a robust decision that finds the best possible result in the worst case. The choice between these models and the principle

of maximum expected utility is more of a human-value judgment rather than a comparison between rewards because all these are models for different ways an agent can be rational. Which way is the best isn't something quantifiable because a selection is based on the belief one model is more rational than the other, which is a human choice. Therefore, we will stick with the standard model of rationality of maximizing expected utility.

3. RESULTS

This section covers the experimental validation of our optimal strategy followed by theoretical results about the geometric structure of the problem and solution.

3.1 Experiment

3.1.1 Data Collection

For the experiment, three different arrangements of robots, destinations, checkpoints, and failure rates are considered. For each arrangement, we compare the path given by our optimal strategy contrasted by strategies where either the robot always heads in the direction of d_1 or d_2. We will run a $100{,}000$-independent-trial, software simulation where robots fail randomly along the way according to our failure model and robots get an opportunity to change their destination and trajectory at communication checkpoints. Under the same conditions for each trial, each strategy will receive a reward at task completion, and a moving, **cumulative average** of total scores over trials is stored.

We plot the moving average over trials for each strategy to measure the expected reward each strategy converges to. Converging at higher rewards means the strategy does better in the long run. Additionally, the theoretical expected value will be plotted as a horizontal line alongside the moving averages as a value of reference of the optimal expected, achievable value. We expect the moving average of the optimal strategy's reward will converge to this theoretically computed expected reward.

3.1.2 Results

The first two experiments showcase the single checkpoint case and the third experiment will be a double checkpoint case.

For the first experiment, the locations as shown in Figure 5 of the r_1, r_2, d_1, and d_2 are at $(0, 10)$, $(10, 10)$, $(-10, 0)$, and $(10, 0)$ respectively. $M(d_1) = 10$ and $M(d_2) = 15$. The checkpoint is 5 units of time. Finally, both the failure rates are $\frac{\log 2}{5}$.

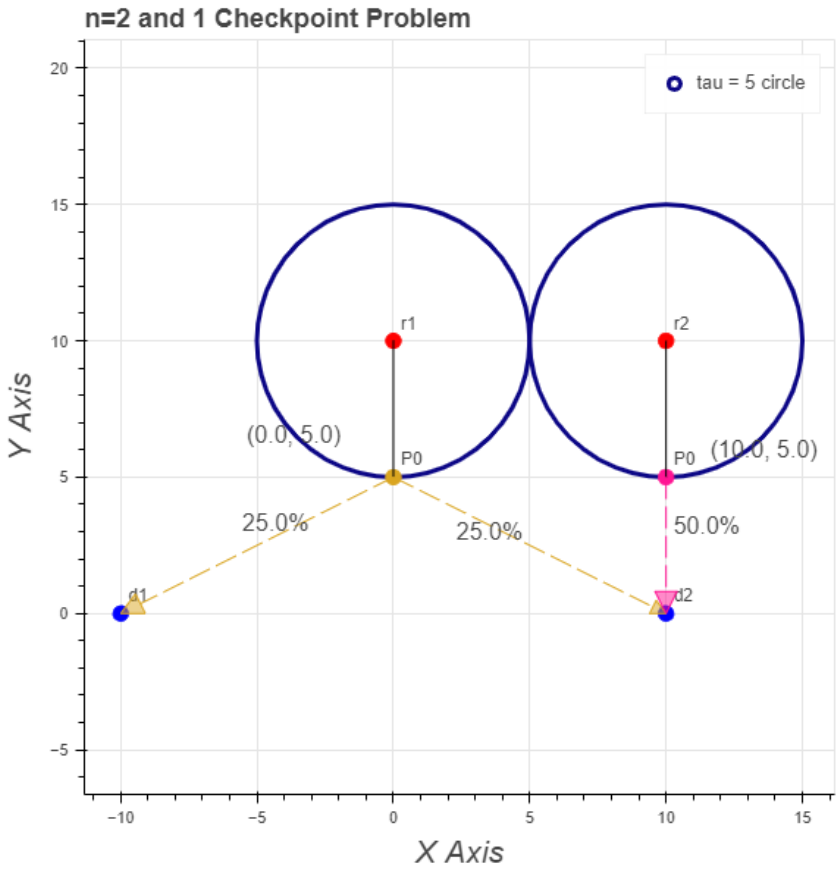

Figure 5: The $P0$s for each robot show the optimal coordinates to move to maximize expected utility. The dashed arrows show the assignment and the percentage of time the robot will follow it. The percentages are derived by flipping 50-50 coins. For example, r_1 will go to d_2 when r_2 dies and r_1 lives. This is the probability two fair coins flip head and tail — 25.0%.

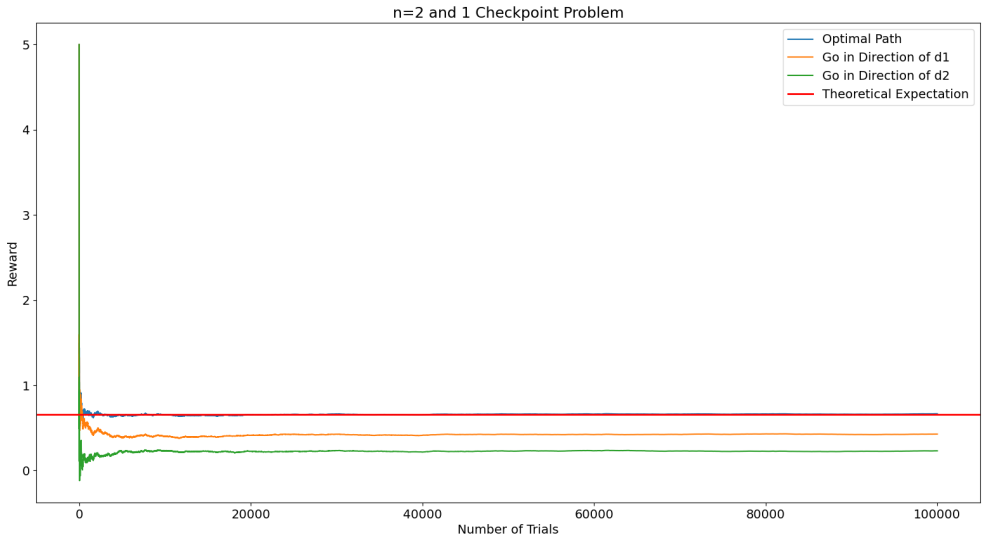

Figure 6: Shows the running average-reward over trials plot. The average reward from following the optimal path converges to the theoretical expected value.

For the second experiment, the locations as shown in Figure 7 of the r_1, r_2, d_1, and d_2 are at $(7, 10)$, $(7, 10)$, $(-10, 0)$, and $(10, 0)$ respectively. $M(d_1) = 10$ and $M(d_2) = 15$. The check-in point is 5 units of time. Finally, the failure rate for r_1 is $\frac{\log 10}{5}$ and the failure rate for r_2 is $\frac{\log 5}{5}$.

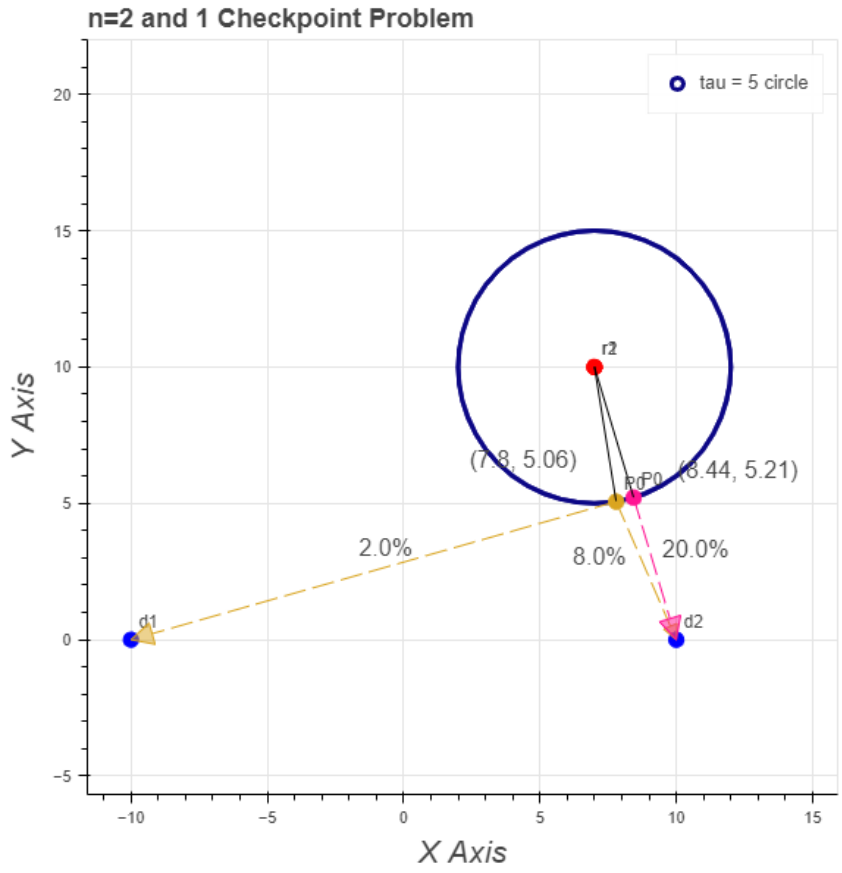

Figure 7: The golden point $P0$ is the optimal coordinate for r_1 to move. The pink point $P0$ is the optimal coordinate for r_2 to move. The dashed arrows show that 2.0% of the time r_1 will go to d_1. But the 8.0% of the time r_2 fails and r_1 lives, r_1 will switch to d_2.

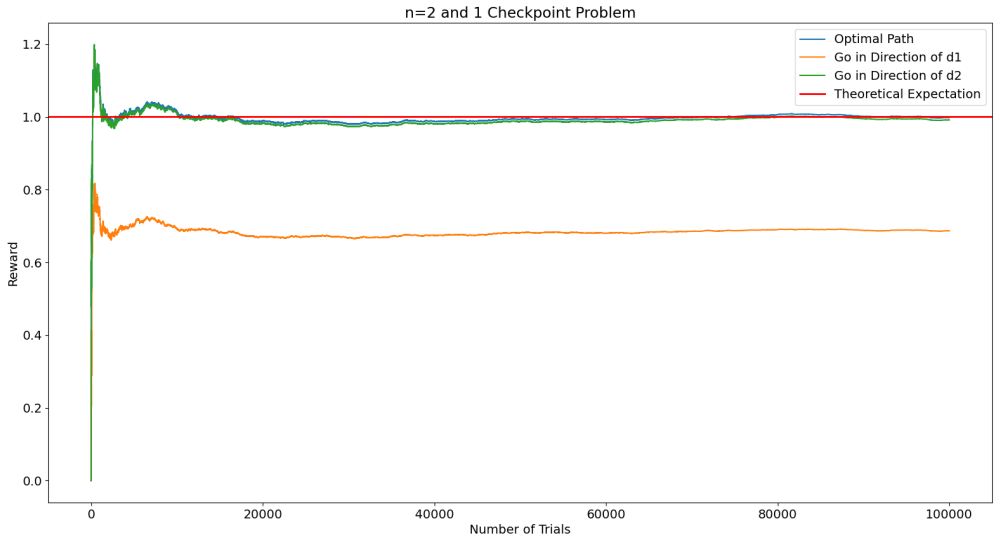

Figure 8: Shows the running average-reward over trials plot. The average reward from following the optimal path converges to the theoretical expected value.

Figures 6 and 8 show in both experiments that the running average-reward of following the optimal path converges to the theoretically estimated value. This validates our methodology for correctness as the Law of Big Numbers says that the average of many trials should converge to the expectation. The running average of going in the direction of d_2 in Figure 8 is close to the theoretically expected reward, but will never converge to it as it is not the optimal direction for r_1 to move in.

Finally, in experiment 3, we validated the theoretical result for two checkpoints. All the conditions shown in Figure 9 are the same as experiment 1 except for the checkpoint times and rate of failures. There are two checkpoint times at $\tau_1 = 3$ and $\tau_2 = 6$. r_1's rate of failure is $\frac{\log 2}{6}$ and r_2's rate of failure is $\frac{\log 10/3}{6}$.

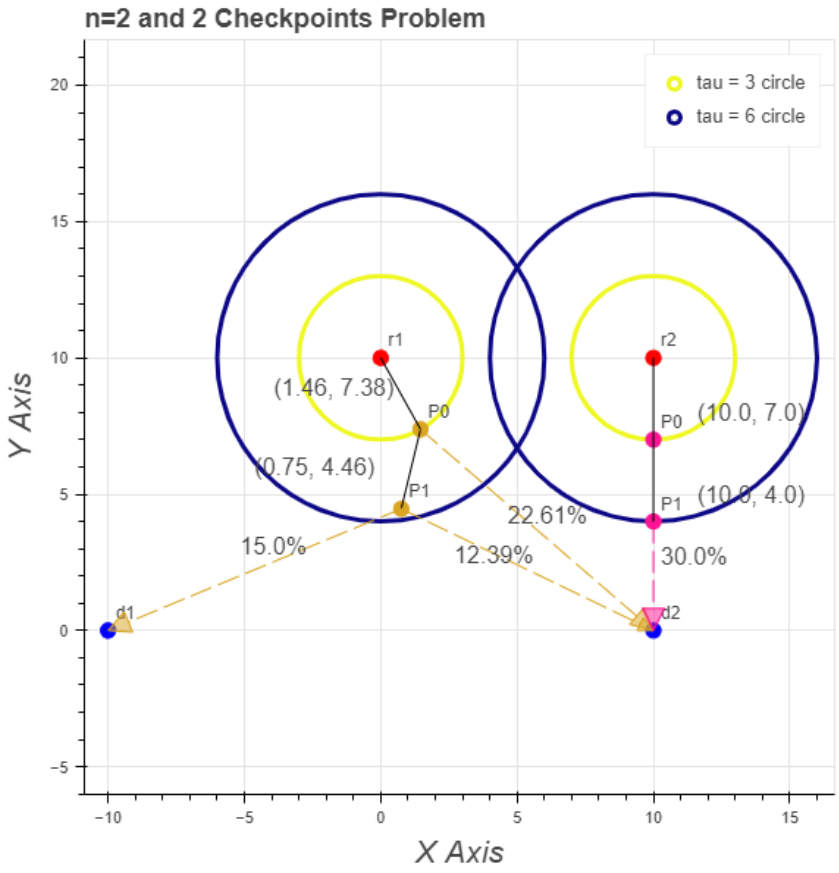

Figure 9: The line segments joined by points $P0$ and $P1$ show the optimal path each robot should move to maximize the expected reward. The dashed arrow at the golden point $P0$ shows r_1 will move to d_2 22.61% of the time it discovers r_2 failed at $\tau = 3$. The arrows at the golden point $P1$ show the possible cases at $\tau = 6$ if both robots were still alive at $\tau = 3$. r_2 never switches from its destination at any point in time.

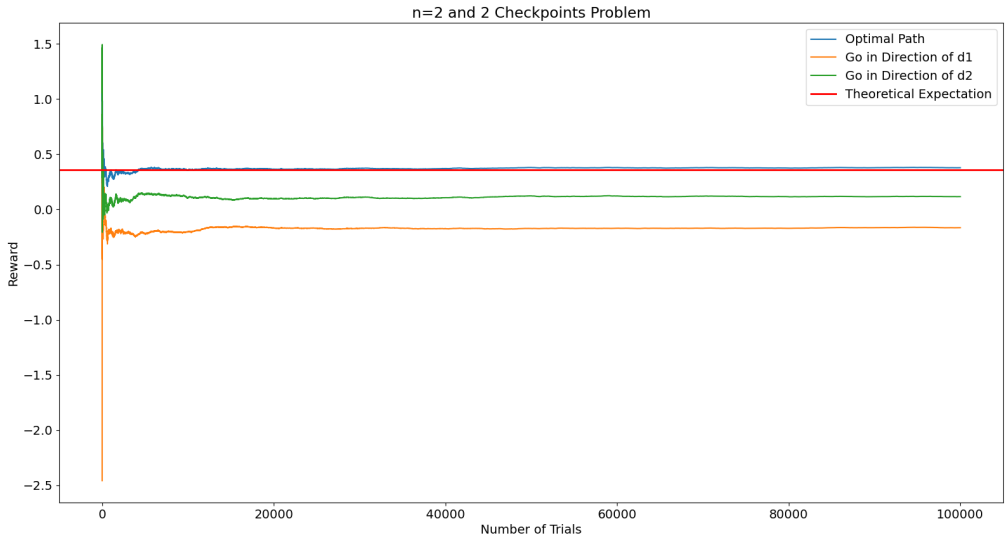

Figure 10: Shows the running average-reward over trials plot. The average reward from following the optimal path converges to the theoretical expected value.

Figure 10 shows that for the two-checkpoint case, the running average converges to the optimal, theoretically-expected value, validating that our methodology generalizes to more than one checkpoint.

3.2 Theoretical Results

Augmenting our numerical approach, we provide geometric explanations and observations for several properties of the problem. The following five topics describe them — the first four about the single check-in case and the last on the k check-in case:

1. Which destination should a robot go to — introduction to the one-sided, reward hyperbola.
2. When is switching relevant to the problem?
3. How do you find the robot that never switches?
4. When do we prefer one strategy over another?
5. Experimental observations on robot paths for k checkpoints.

3.2.1 Which Destination Should a Robot Go to?

Suppose r_2 is the robot that never switches. Given any position q on the 2-D plane that r_1 finds itself in, should it go to d_1 or d_2? Explicitly, for which q should r_1 go to d_2 over d_1 to get more reward? The following equation compares the expected reward between the two decisions:

$$p_1\,\psi(d_1, ||d_1 - q|| + \tau) < p_1\left(p_2\,\psi(d_1, ||d_1 - q|| + \tau) + (1 - p_2)\,\psi(d_2, ||d_2 - q|| + \tau)\right) \tag{16}$$

Deriving the end result, a well-studied form of an equation is found:

$$||d_2 - q|| - ||d_1 - q|| < \phi^{-1}(M(d_2) - M(d_1)) \tag{17}$$

This is the equation for the hyperbola, specifically a one-sided one, where d_1 and d_2 are the foci and the right-hand side of the equation is a constant. This equation says that you should go to the destination that's on the same side of the hyperbola as your location q because the relative reward is higher. Let this be known as the **hyperbola property**.

This hyperbola is important because it says a lot about the relationship between rewards and distances and is used to explain subsequent interesting properties of the problem. Assume for simplicity that $\phi(t) = t$. An example is shown in Figure 11

For example, if you constrain q to be the set of points on the arc of the τ-circle around r_1, each section of the arc, that is divided by the hyperbola, prescribes which destination the robot should go to to get the most reward.

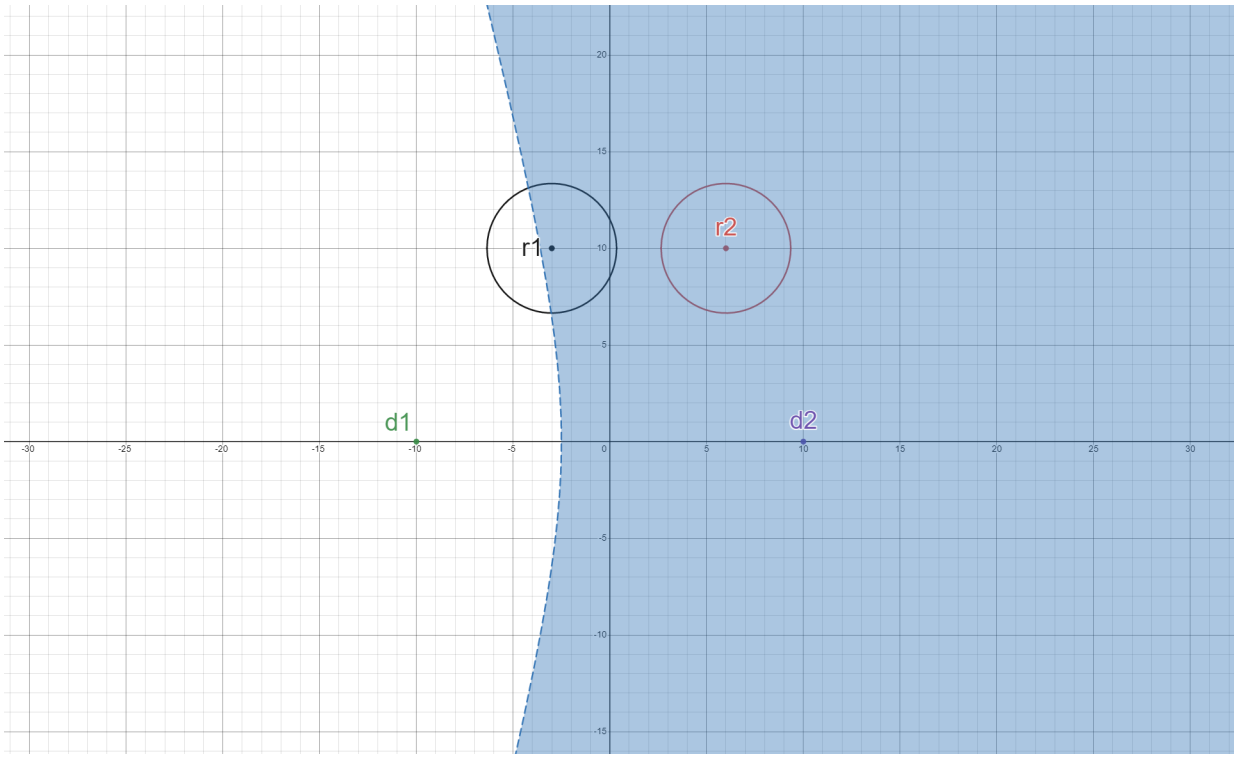

Figure 11: A simple example of a one-sided hyperbola for equation 17 when $M(d_1) = 10$ and $M(d_2) = 15$. The shaded region is the set of all points where the equation holds true. $\tau = 3.35$-circles are being shown around the robots.

Additionally, if q is exactly on the hyperbola line, then the robot is indifferent to either destination — going to either one will give the same reward.

Finally, if the absolute value of the difference in reward is greater than the distance between the two destinations — $|M(d_2) - M(d_1)| > ||d_2 - d_1||$ — then no matter where the robot is on the plane, there is only one destination that will give more reward, as shown in Figure 12.

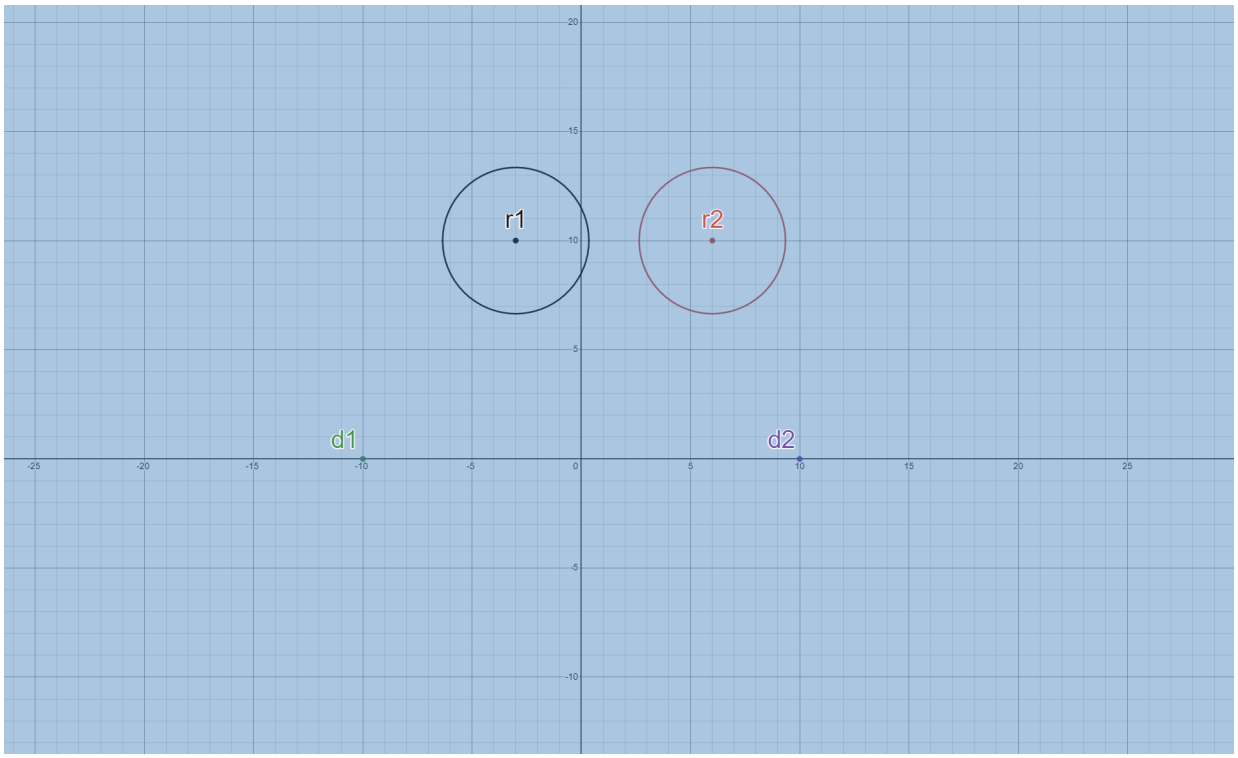

Figure 12: An example of a one-sided hyperbola when $M(d_1) = 10$ and $M(d_2) = 30$. Because the difference in reward is 20, the same as the distance between d_1 and d_2, all the points are shaded. This indicates no matter where the robot starts or ends up at, moving to d_2 will always have more reward than going to d_1.

3.2.2 When is Switching Relevant to the Problem?

We can determine if a robot could potentially switch assignments by the starting locations of the two robots.

Lemma 3.2.1. *If the starting positions of both robots are on opposite sides of the hyperbola, then no switching will occur.*

Proof. Neither robot will ever go to a destination on the opposite side of the hyperbola from itself; both will remain on their own side. When both are alive, if they both go to the opposite side, their reward will be sub-optimal compared to staying on the same side by the **hyperbola property**. More generally, robots will be losing out on reward every time they cross the hyperbola boundary

to go to another destination. If they choose to cross back, then they wasted even more potential reward. Because there is no merit for a robot to cross boundaries and each robot has its own unique destination to go to, the optimal policy will never have switching. □

Lemma 3.2.2. *If the starting positions of both robots are on the same side of the hyperbola, then switching might occur.*

Proof. Let d_2 be the destination on the same side as the robots. We know by Lemma 2.1.3 that one robot will never switch so choose r_2 as the robot that never switches.

Case 1: r_2 goes to d_1. Then since r_1 is on the same side as d_2, by the **hyperbola property** it should always go to d_2. So no switching occurs. This is the uninteresting case.

Case 2: r_2 goes to d_2. Suppose the case when both robots are alive by time τ. Then by the restriction of task assignment, r_1 must go to d_1. Suppose the case when r_2 fails. Then r_1 could go to d_2 or it could stick with going to d_1. Unlike the previous proofs, there is no statement saying one decision is always optimal so switching from d_1 to d_2 on r_2's failure might occur depending on the situation. □

Lemma 3.2.3. *If the starting positions of both robots are on the same side of the hyperbola, and the τ-circles of both robots never intersect the hyperbola, then switching must occur.*

Proof. Assume the same conditions as Lemma 3.2.2. Because the circle of reachable points is always on one side of the hyperbola, no matter where r_1 goes, r_1 should always switch to d_2 on r_2's failure by the **hyperbola property**. □

3.2.3 How do you Find the Robot that Never Switches?

It is important to find the robot that never switches, as this simplifies the problem by reducing it to optimizing the policy of the other robot. We will focus on the situation when both robots are on the same side of the hyperbola since otherwise, neither will switch by Lemma 3.2.1.

Lemma 3.2.4. *Assume the starting positions of both robots are on the same side of the hyperbola. Let q_1 and q_2 be the possible locations on r_1 and r_2's τ-circle respectively. Then if for all q_1 and q_2, $||d_2 - q_1|| - ||d_1 - q_1|| < ||d_2 - q_2|| - ||d_1 - q_2||$ is true, then r_2 is the robot that never switches.*

Proof. Assume the case when both robots live. There are two assignments — r_1 and r_2 go to d_1 and d_2 respectively or swapped. Then the comparison that the former assignment's reward is greater than the latter's is the following inequality:

$$p_1\, p_2 \left(\psi(d_1, ||d_1 - q_1|| + \tau) + \psi(d_2, ||d_2 - q_2|| + \tau)\right) < p_1\, p_2 \left(\psi(d_2, ||d_2 - q_1|| + \tau) + \psi(d_1, ||d_1 - q_2|| + \tau)\right) \tag{18}$$

The inequality simplifies to the assumption:

$$||d_2 - q_1|| - ||d_1 - q_1|| < ||d_2 - q_2|| - ||d_1 - q_2|| \tag{19}$$

Conversely, if the assumption is true, then r_1 will always go to d_1 and r_2 will always go to d_2 for all q_1 and q_2 when neither robots fail. Then the only way to make r_2 switch assignments is if r_1 fails, then r_2 goes to d_1. Because the **hyperbola property** says this is sub-optimal to staying on the same side of the hyperbola going to d_2, and r_2 goes to d_2 anyways when both robots are alive, then r_2 always goes to d_2. r_2 must be the robot that never switches. □

If the strict converse of the inequality is true, then we can swap the labels of the robots to achieve the same effect. In section 3.3, we will address the scenario in which the assumption that the inequality holds for all q_1 and q_2 is violated.

3.2.4 When do we Prefer One Strategy Over Another?

Suppose d_2 and both robots are on the same side of the hyperbola, r_2 is the robot that never switches from d_2, and the τ-circle of r_1 intersects with the hyperbola. Then by Lemma 3.2.2, r_1 has to choose between two strategies — "Never-Switch" or "Switch-If-Needed." The numerical approach is to solve each objective functions 11 and 12 and take the argmax between the two angles for the optimal direction.

Now, we would like to know at what probabilities of living, p_1 and p_2, does the optimal strategy change from "Never-Switch" to "Switch-If-Needed." For example, we know that when

the probability of $p_2 = 1 - r_2$ never dies — r_1 will always move in a straight path to d_1 by the task assignment restriction. When $p_2 = 0$ — r_2 will die — r_1 will move in a straight path to d_2 by the **hyperbola property**. From observation, we know the optimal angle, θ_1^*, and its corresponding strategy continuously changes from one to the other as p_2 is monotonically perturbed. We would like to know at what number p_2 the strategies change and if the change in θ_1^* is smooth or discontinuous along the arc.

First is the relevance of probability p_1. p_1 is a scalar for equations 11 and 12. Therefore, they have no bearing on the optimization unless $p_1 = 0$ in which there is no policy to be optimized since r_1 will always die. Assume from now on that p_1 is a fixed, positive number to keep things simple.

Second, we would like to know how θ_1^* is perturbed as we change p_2. We do this by showing a discontinuous property of θ_1^* and relate that to p_2's perturbation.

Lemma 3.2.5. *Let P be the point of intersection between r_1's τ-circle and the line segment $\overline{r_1 d_1}$. If P resides in the arc segment that's on the same side as d_1, then going to P and never switching gets the most reward out of all the other points on that arc segment.*

Proof. Point P is the orthogonal projection from r_1 to d_1 so the time it takes to move from P d_1 is the smallest. Additionally, by the **hyperbola property**, any other point on the arc segment also follows the "Never-Switch" strategy. Then since going to P takes the least time for the same reward taken at d_1, P is the optimal location to move to. □

This implies that θ_1^* is inherently discontinuous as p_2 is perturbed. θ_1^* will never move continuously across the whole arc from one side to another. At the very least, once it crosses the hyperbola from one arc segment to the other will the strategy and angle change to where P should be. An even stronger statement on this discontinuity is made:

Lemma 3.2.6. *Let X be the point of intersection between r_1's τ-circle and the hyperbola that also lies on the arc between P and Q, the orthogonal projections to the arc from r_1 to either destination. There exists a section of the arc on the d_2 side of the hyperbola that continuously extends from X*

such that doing the "Switch-If-Needed" strategy there has less reward than the "Never-Switch" strategy at P.

Proof. First I must show that the reward as θ_1^* moves from one side of the hyperbola to another is continuous. Because the objective functions for each arc segment are continuous, the rewards are continuous. Next, I must show that the reward is continuous at the point these two arcs meet: X. Because the robot is indifferent to the destination it goes to when it's exactly on the hyperbola, the rewards of doing either strategy at X must be the same. Therefore, the reward is continuous as θ_1^* moves across the whole arc.

Next, the reward at X of doing either strategy, U_X is strictly less than the reward of the "Never-Switch" strategy at P, U_P. By Lemma 3.2.5, X is included on the arc segment on the same side of d_1 as that point is indifferent, therefore $U_X < U_P$.

Finally, by the intermediate value theorem, there must be some continuous segment of points as θ_1^* moves past X onto the side of d_2 where the reward is still less than or equal to U_P. Therefore, there exists a segment of the arc on the same side of d_2 where "Switch-If-Needed" is sup-optimal to "Never-Switch" at P. □

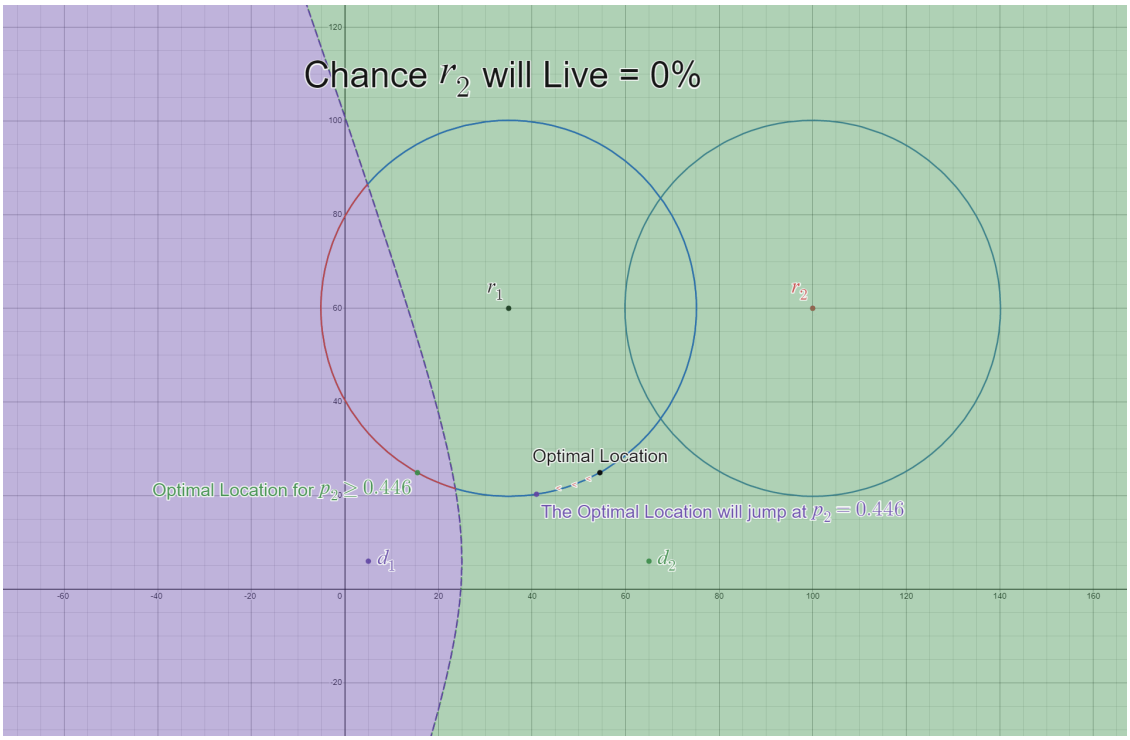

Figure 13: The animation depicts the optimal-location point r_1 should go to as p_2 monotonically increases from 0 to 1. As p_2 increases, before the point reaches the hyperbola line, it will 'jump' to the shaded purple region. The green region is where r_1 should go to d_2 and the purple region is for d_1. The red arrows indicate the direction the optimal point will move along the arc as p_2 increases.

The direct implication of Lemma 3.2.6 is that there exists a threshold of p_2, as it is monotonically perturbed up from 0, where simply committing to d_1 without switching is better than switching. Because the θ_1^* associated with that threshold does not continuously cross the hyperbola but jumps to the other side, it is interesting to see this non-linearity.

3.2.5 Experimental Observations on Robot Paths for k Checkpoints.

An observation to note for some specific problem arrangements is that the optimal path seems to curve. For example, the locations as shown in Figure 14 of the r_1, r_2, d_1, and d_2 are at $(0, 10)$, $(10, 10)$, $(-10, 0)$, and $(10, 0)$ respectively. $M(d_1) = 10$ and $M(d_2) = 15$. The checkpoints are $2, 3, \frac{4}{5} * 6$, and 6 units of time. r_1's rate of failure is $\frac{\log 2}{6}$ and r_2's is $\frac{\log 10/3}{6}$. This is

probably due to r_1 having more opportunities to switch to d_2 at the beginning but fewer as time passes.

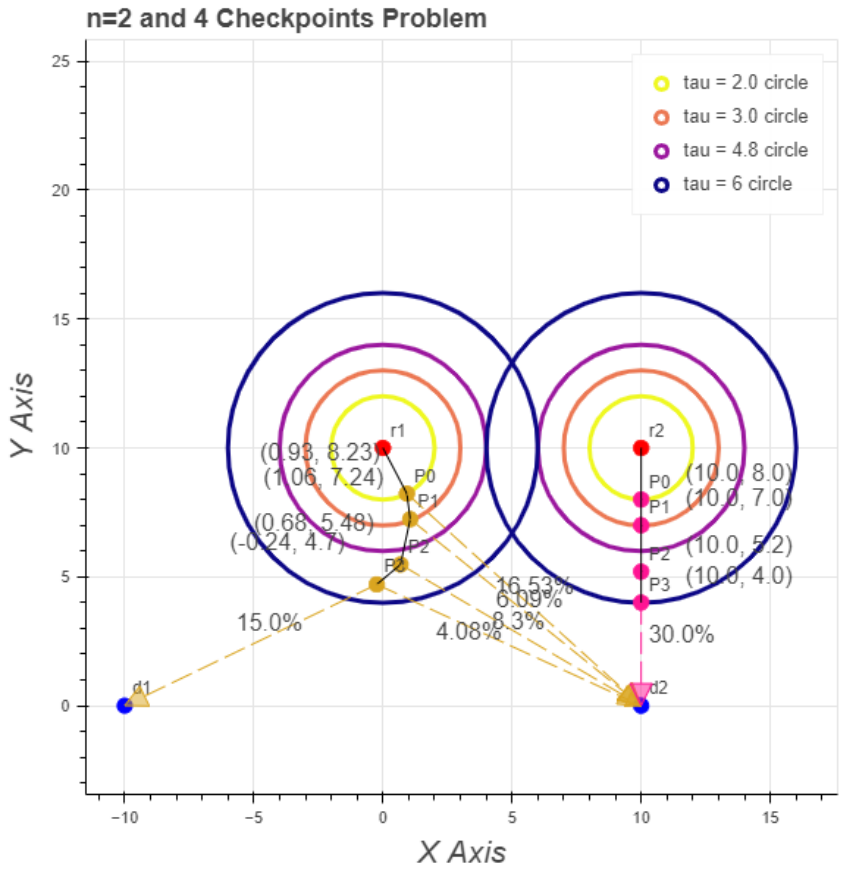

Figure 14: There are 4 checkpoints for the robots to communicate. The line segments for r_1 seem to curve toward d_1. The points Px do not necessarily have to be on the τ-circles, but as noted in Lemma 2.1.2.

3.3 Questions Left Partially Unanswered

Some scenarios and conditions that we have not fully addressed are:

1. What is the relationship between differing rates of robot failure to task assignment?
2. Caveat to finding the robot that never switches.

3.3.1 What is the Relationship between Differing Rates of Robot Failure to Task Assignment?

We mentioned in Lemma 3.2.4 about the scenario when for all q_1 and q_2 the assumption for equation 19 fails to hold. We will talk about why this scenario is important and related to the probability of failures. In the following examples, assume the conditions of Lemma 3.2.4 except for equation 19 are true.

Starting with the simple case, assume that the robots start in the same location, $r_1 = r_2$, and their probabilities to live are different, $p_1 > p_2$. We know one of them never switches from d_2. Intuitively, the robot with the greater chance of living should go to d_2 as it would claim more reward on average. But if we slightly perturb the location of r_1 to be farther away from d_2, does the fact that r_1's probability to live still matter?

Intuitively, it seems that the robot with the higher probability should be the one to never switch, but so far this has never been addressed. We can determine a way to address this utilizing equation 19.

Equation 19 is an equation of 2 variables, q_1 and q_2, so it is hard to imagine this in a 2D plane. But if the right-hand side were a constant, then the equation satisfies the definition of a hyperbola! To visualize this, we treat q_1 as any point on the plane and instantiate q_2 to be some point on r_2's τ-circle. But what point should we instantiate q_2 to be?

The answer is not a single point, but multiple points. Specifically, we pretend for a moment that r_1 is the robot that never switches and solve the optimal policy for each strategy — objective functions 13 and 14 — giving us two candidates for q_2, an example shown in Figure 15. We know that q_2 must always be either one of these. This reduces q_2 from an infinite amount of points to just two points. By plugging these into equation 19 to form a hyperbola, we could do an analysis similar to what we did with the reward hyperbola. This observation is partially unaddressed because we ran out of time to do so.

The reason why the rate of failures is relevant is because one of the points, the solution to equation 13, is a function of the probability p_1. So the hyperbola where swapping which robot never switches is dependent on the probabilities of the problem.

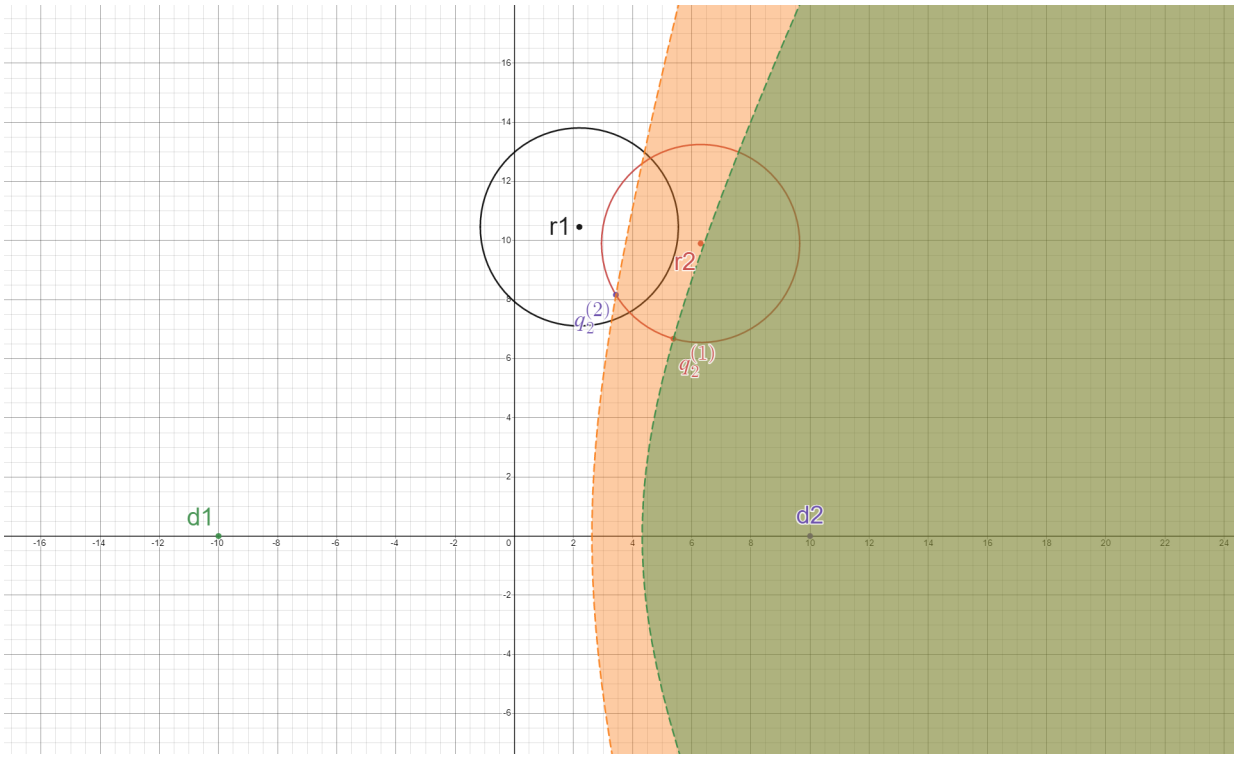

Figure 15: The two candidates for q_2 are shown with their associated hyperbola — $q_2^{(2)}$ instantiates the orange hyperbola and $q_2^{(1)}$ instantiates the green hyperbola. The shaded region is where if r_1 moved to, then when both robots are alive r_1 will move to d_2 and r_2 will move to d_1. $q_2^{(1)}$ is the point r_2 would go to if it was implementing the "Switch-If-Needed Swapped" strategy with $p_1 = 0.5$. Here, $M(d_1) = 10$ and $M(d_2) = 15$.

3.3.2 Caveat to Finding the Robot that Never Switches

Assuming the conditions of the previous section, we know the robot that never switches is the one whose task assignment in both the robot's alive cases is d_2. The prior section tells us that figuring this out is dependent on the probabilities of the robot if the assumption for Lemma 3.2.4 does not hold. We could simply numerically solve all four objective functions to find out the robot that never switches, but that defeats the purpose of making a stronger statement about the problem behavior based on its inputs. Because our analysis of the hyperbola based on equation 19 is incomplete, we have only partially addressed this answer.

4. CONCLUSION

This thesis presents an approach to solve the assignment and trajectory planning problem, for two robots with sporadic communication and risk of failures, that maximizes the expected reward. We introduced the communication and failure models robots are subject to and argued for ways we could simplify the problem. We showed for two robots and two destinations, a calculus approach to solving the problem and explained why we used numerical methods over an analytical approach. After that, we validated our theoretical methodology with experiments, showing with our simulation that the moving average-reward from following the optimal policy converges to the theoretically, expected reward. Finally, we delved into the geometric properties of the problem to explain certain properties such as 'jumping' and 'switching,' utilizing the one-sided, reward-difference hyperbola as a tool. In the end, we left with some closing statements on certain properties we observed through experimentation but were unable to prove.

There are more questions to be asked, such as when are the optimal times to communicate or how do we solve this problem for more than two robots? Although we were unable to answer these, we believe that this thesis lays a foundation for future work tasked with generalizing beyond the scope of our problem. We hope that this inspires further progress to be made in incorporating proactive planning to robotics problems.

REFERENCES

[1] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin, and P. Corke, "Data collection, storage, and retrieval with an underwater sensor network," in *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, SenSys '05, (New York, NY, USA), p. 154–165, Association for Computing Machinery, 2005.

[2] A. Zolich, D. Palma, K. Kansanen, K. Fjørtoft, J. Sousa, K. H. Johansson, Y. Jiang, H. Dong, and T. A. Johansen, "Survey on communication and networks for autonomous marine systems," *Journal of Intelligent & Robotic Systems*, vol. 95, pp. 789–813, Sep 2019.

[3] M. Turpin, N. Michael, and V. Kumar, "Concurrent assignment and planning of trajectories for large teams of interchangeable robots," in *2013 IEEE International Conference on Robotics and Automation*, pp. 842–848, 2013.

[4] L. Liu and D. A. Shell, "Assessing optimal assignment under uncertainty: An interval-based algorithm," *The International Journal of Robotics Research*, vol. 30, p. 936–953, Jun 2011.

[5] C. Nam and D. A. Shell, "Robots in the huddle: Upfront computation to reduce global communication at run time in multirobot task allocation," *IEEE Transactions on Robotics*, vol. 36, no. 1, pp. 125–141, 2020.

[6] T. Olsen, N. M. Stiffler, and J. M. O'Kane, "Robust-by-design plans for multi-robot pursuit-evasion," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 10716–10722, 2022.

[7] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 2681–2690, PMLR, 06–11 Aug 2017.

[8] S. J. Russell and P. Norvig, *Artificial Intelligence: A modern approach*. Pearson Education Limited, 4th ed., 2020.

APPENDIX: Analytical Approach

First, convert the problem into polar coordinates. Convert d_1, d_2, q, r into polar coordinates. To simplify the process:

1. Translate everything so r is at the origin.
2. Rotate everything so that d_1 is 90° to the origin.

Define these variables after the transformations:

- $l_1 = ||d_1 - r||$
- $l_2 = ||d_2 - r||$
- $\theta_1 = \frac{\pi}{2}$
- $\theta_2 = \arctan(d_2.y, d_2.x)$
- $d_1 = (l_1, \theta_1)$
- $d_2 = (l_2, \theta_2)$
- $q = (\tau, \theta)$
- $r = (0, 0)$

The general formula for the distance between two polar coordinates $(x, \theta_1), (y, \theta_2)$ is

$$D = \sqrt{x^2 + y^2 - 2xy\cos(\theta_1 - \theta_2)}.$$

Therefore the distances between q and each destination are:

$$||d_1 - q|| = \sqrt{l_1^2 + \tau^2 - 2l_1\tau\cos(\theta_1 - \theta)},$$

and

$$||d_2 - q|| = \sqrt{l_2^2 + \tau^2 - 2l_2\tau\cos(\theta_2 - \theta)}.$$

The objective function to solve, for arbitrary probability weights w_1 and w_2, is then

$$\underset{\theta}{\text{minimize}}\ w_1\sqrt{l_1^2 + \tau^2 - 2l_1\tau\cos(\theta_1 - \theta)} + w_2\sqrt{l_2^2 + \tau^2 - 2l_2\tau\cos(\theta_2 - \theta)}.$$

We take the derivative and set it equal to 0 to solve for the critical points.

$$\begin{aligned}
\frac{\mathrm{d}f}{\mathrm{d}\theta} =& w_1\frac{-l_1\tau\sin(\theta_1 - \theta)}{\sqrt{l_1^2 + \tau^2 - 2l_1\tau\cos(\theta_1 - \theta)}} + w_2\frac{-l_2\tau\sin(\theta_2 - \theta)}{\sqrt{l_2^2 + \tau^2 - 2l_2\tau\cos(\theta_2 - \theta)}} = 0 \\
& w_1\frac{-l_1\tau\sin(\theta_1 - \theta)}{\sqrt{l_1^2 + \tau^2 - 2l_1\tau\cos(\theta_1 - \theta)}} = w_2\frac{l_2\tau\sin(\theta_2 - \theta)}{\sqrt{l_2^2 + \tau^2 - 2l_2\tau\cos(\theta_2 - \theta)}} \\
& \frac{\sqrt{l_2^2 + \tau^2 - 2l_2\tau\cos(\theta_2 - \theta)}}{\sqrt{l_1^2 + \tau^2 - 2l_1\tau\cos(\theta_1 - \theta)}} = -\frac{w_2 l_2\sin(\theta_2 - \theta)}{w_1 l_1\sin(\theta_1 - \theta)} \\
& \frac{l_2^2 + \tau^2 - 2l_2\tau\cos(\theta_2 - \theta)}{l_1^2 + \tau^2 - 2l_1\tau\cos(\theta_1 - \theta)} = (\frac{w_2 l_2\sin(\theta_2 - \theta)}{w_1 l_1\sin(\theta_1 - \theta)})^2.
\end{aligned}$$

Recall the sin and cos angle difference formulas and what happens when one of the angles is π.

$$\sin(\theta_1 - \theta_2) = \sin\theta_1\cos\theta_2 - \cos\theta_1\sin\theta_2.$$

$$\cos(\theta_1 - \theta_2) = \cos\theta_1\cos\theta_2 + \sin\theta_1\sin\theta_2.$$

$$\sin(\frac{\pi}{2} - \theta_2) = \cos(\theta_2).$$

$$\cos(\frac{\pi}{2} - \theta_2) = \sin(\theta_2).$$

Let's substitute this identity for trigs with θ_1.

$$\frac{l_2^2+\tau^2-2l_2\tau\cos(\theta_2-\theta)}{l_1^2+\tau^2-2l_1\tau\sin(\theta)}=(\frac{w_2l_2\sin(\theta_2-\theta)}{w_1l_1\cos(\theta)})^2.$$

$$(l_2^2+\tau^2-2l_2\tau\cos(\theta_2-\theta))\,(w_1l_1\cos(\theta))^2=(l_1^2+\tau^2-2l_1\tau\sin(\theta))\,(w_2l_2\sin(\theta_2-\theta))^2.$$

$$(l_2^2+\tau^2-2l_2\tau\cos(\theta_2-\theta))\,(w_1l_1\cos(\theta))^2-(l_1^2+\tau^2-2l_1\tau\sin(\theta))\,(w_2l_2\sin(\theta_2-\theta))^2=0.$$

Unfortunately, the simplification becomes ugly.

$$\begin{aligned}
&2\,\tau\,l_1\,{l_2}^2\,{w_2}^2\cos(\theta_2)^2\sin(\theta)^3\\
&-4\,\tau\,l_1\,{l_2}^2\,{w_2}^2\,\cos(\theta_2)\,\sin(\theta_2)\,\sin(\theta)^2\,\cos(\theta)\\
&-{l_2}^2\,{w_2}^2\,(\tau^2+{l_1}^2)\cos(\theta_2)^2\sin(\theta)^2\\
&+\left(2\,\tau\,l_1\,{l_2}^2\,{w_2}^2\sin(\theta_2)^2-2\,\tau\,{l_1}^2\,l_2\,{w_1}^2\,\sin(\theta_2)\right)\sin(\theta)\,\cos(\theta)^2\\
&+2\,{l_2}^2\,{w_2}^2\,\cos(\theta_2)\,\sin(\theta_2)\,(\tau^2+{l_1}^2)\,\sin(\theta)\,\cos(\theta)\\
&-2\,\tau\,{l_1}^2\,l_2\,{w_1}^2\,\cos(\theta_2)\,\cos(\theta)^3\\
&+\left({l_1}^2\,{w_1}^2\left(\tau^2+{l_2}^2\right)-{l_2}^2\,{w_2}^2\sin(\theta_2)^2\,(\tau^2+{l_1}^2)\right)\cos(\theta)^2.
\end{aligned}$$

Collect all the coefficients that are not a function of θ from top to bottom as a, b, c, d, e, f, g. Since these are truly numeric values, we can compute them to make them more compact.

$$0=a\sin(\theta)^3+b\sin(\theta)^2\cos(\theta)+c\sin(\theta)^2+d\sin(\theta)\cos(\theta)^2+e\sin(\theta)\cos(\theta)+f\cos(\theta)^3+g\cos(\theta)^2.$$

To turn the equation computer solvable, we need to do a tangent half-angle substitution and introduce a new variable t. This will turn the equation into a single variable.

$$t=\tan(\frac{\theta}{2}).$$

$$\sin(\theta)=\frac{2t}{1+t^2}.$$

$$\cos(\theta) = \frac{1 - t^2}{1 + t^2}.$$

By substituting and simplifying, we get

$$\frac{\begin{array}{r}(g - f)\,t^6 + (2\,d - 2\,e)\,t^5 + (4\,c - 4\,b + 3\,f - g)\,t^4 + (8\,a - 4\,d)\,t^3 \\ + (4\,b + 4\,c - 3\,f - g)\,t^2 + (2\,d + 2\,e)\,t + f + g\end{array}}{t^6 + 3\,t^4 + 3\,t^2 + 1} = 0.$$

The denominator factored is

$$t^6 + 3\,t^4 + 3\,t^2 + 1 = (t^2 + 1)^3.$$

$$t^2 + 1 = 0 \implies t^2 = -1.$$

This is impossible therefore the denominator will never be 0. So we can cancel it out.

$$\begin{array}{r}(g - f)\,t^6 + (2\,d - 2\,e)\,t^5 + (4\,c - 4\,b + 3\,f - g)\,t^4 + (8\,a - 4\,d)\,t^3 \\ + (4\,b + 4\,c - 3\,f - g)\,t^2 + (2\,d + 2\,e)\,t + f + g\end{array} = 0.$$

We are unable to progress any further with this analytical expression because it is a polynomial of degree six and there is no general closed-form formula for the roots. Our next step is to resort to numerical methods to solve the minimization problem.