ASSIGNMENT AND PURSUIT IN TEMPORALLY HETEROGENEOUS ROBOTIC TEAMS

A Thesis
Presented to
The Academic Faculty

by

Marius Oei

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Mechanical Engineering

Georgia Institute of Technology
August 2016

ASSIGNMENT AND PURSUIT IN TEMPORALLY HETEROGENEOUS ROBOTIC TEAMS

Approved by:

Professor Aaron Ames, Committee Chair
School of Mechanical Engineering
Georgia Institute of Technology

Professor Magnus Egerstedt, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Oliver Sawodny
Institute for System Dynamics
University of Stuttgart

Professor Cristina Tarín
Institute for System Dynamics
University of Stuttgart

Date Approved: 14 July 2016

To my brothers

Yannick and Victor.

You're the best.

ACKNOWLEDGEMENTS

First and foremost, I want to thank my advisor Dr. Magnus Egerstedt for his guidance that made this work possible. He always made sure that this thesis is my own work, while still providing the most valuable input. It has been an honor to be a member of his research lab over the past year.

Further I would like to thank my reading committee for providing their expertise and their time to evaluate my results.

My appreciation goes to Prof. Sawodny and Dr. Neitzel for creating the Stuttgart and Georgia Tech Joint Master's program. They have both put great effort into this program to ensure the success of Benjamin and me, the first students to achieve this joint degree.

I also thank the members of the GRITS-lab, past and present, for fruitful discussions and valuable input. Thank you to Dr. Daniel Pickem for suggesting and explaining the use of Markov chain Monte Carlo methods and to him and Paul Glotfelter for creating the Robotarium. It made the robotic implementation as simple as a walk in the park.

A special thank you goes to my friends Ahmed, Ahmed, Benjamin, Chris, Caroline, Euiyoung, Nadine, Thomas and Xiaoshu for making this year abroad one of the best experiences of my life and always expanding my horizons. I have grown so much in this past year thanks to you.

Finally, I want to express my gratitude to my parents for their never ending support throughout my life and for making me the person that I am today. Without your help, none of this would have been possible. I am greatly in your debt.

Contents

List of Tables

List of Figures

SUMMARY

Traditionally, robotic systems are built to move as fast as possible. In contrast to this, we investigate slowness and its effects on heterogeneous robotic teams inspired by biological systems. An assignment problem for static targets and a team pursuit problem for heterogeneous evaders are addressed. The value of slowness in solving these problems optimally is examined. We further assemble the optimal teams for given problems by finding a compromise between performance and energy consumption or monetary cost. The results are validated in simulation and implemented on a robotic testbed.

Chapter I

INTRODUCTION

Robotic teams are groups of robots that are designed to solve tasks cooperatively. The advent of high performance miniaturized hardware and the maturity of control algorithms have given rise to this recent field of research.

For decades, technical systems have been developed to complete tasks faster and faster. Examples for this can be found in the production of new industrial manipulators [8], the evolution of the automobile [40] or the development of quadrupedal sprinting robots [57]. The advantages of this development are clear: faster execution means being able to execute more tasks in the same time (e.g. assembly tasks), easier satisfaction of time constraints (in search and rescue scenarios) and better interaction with other fast systems (like the control of fast chemical reactions).

In light of these facts, it is remarkable that examples of the complete opposite can be found in nature. The existence of species like the sloth suggests that sometimes it is beneficial to be slow instead of fast, considering that natural selection should have eradicated them otherwise. For the same reason, it appears as if there should be some value or at least no harm in the coexistence of fast and slow animals in an ecosystem. Under some circumstances, slowness can open an ecological niche that is inaccessible to faster lifestyles, leading to better utilization of the available resources.

In this thesis, we will combine these two arguments to make an approach towards heterogeneous robotics with an emphasis on slow participants in networked systems. We investigate two problems: the optimal assignment of robotic agents to target positions with respect to task completion time and energy consumption and a robotic pursuit problem of a heterogeneous team of moving target robots. First, some of the concepts and previous work on the topics are introduced.

1.1 Biologically Inspired Heterogeneity

In nature, heterogeneity – usually referred to as biodiversity – can be found almost everywhere [29, 21, 13]. The fact that heterogeneity is the outcome of millions of years of evolution by natural selection leads us to believe that there has to be an inherent utility in the variation of properties among coexisting species. Further, we can observe heterogeneity even within societies of one species. A prominent example of a species that shows heterogeneity are leaf-cutter ants, which have developed a system of caste and division of labor [58]. Another obvious example are human societies, which have a high grade of separation of tasks leading to individual specialization. In fact, it has been shown that heterogeneity correlates with the fitness of societies [47].

Homogeneity on the other hand is often only introduced by humans [25]. An example for this is the homogenization of species in commercial agriculture, where the focus on a small number of high-performance crops leads to a global reduction in biodiversity [30]. This has been linked to the worsening of pest problems, requiring constant human intervention to protect crops [1].

Even though homogeneity simplifies complex systems and makes them easier to control for humans, heterogeneity seems to have functional advantages. We seek to find out how to make use of these properties by incorporating heterogeneity into robotic teams. While heterogeneity is a very broad term that describes a vast number of variations in parameters, we will focus our attention on temporal heterogeneity. This means that members of the society work on different timescales, some of them being slow while others are faster.

1.2 Introducing Slowness

To establish temporal heterogeneity in a team, there has to be some value of slowness. In particular, instead of trying to reach goals as quickly as possible, we aim to find out in what ways slowness can be beneficial for the completion of a task. In nature, slow behavior can be observed in the tree sloth (depicted in Fig. 1) and the slow loris, as described in [2, 43]. But what is the evolutionary benefit of this slothfulness?

The most apparent advantage of slow movement is the reduction of energy expenditure.

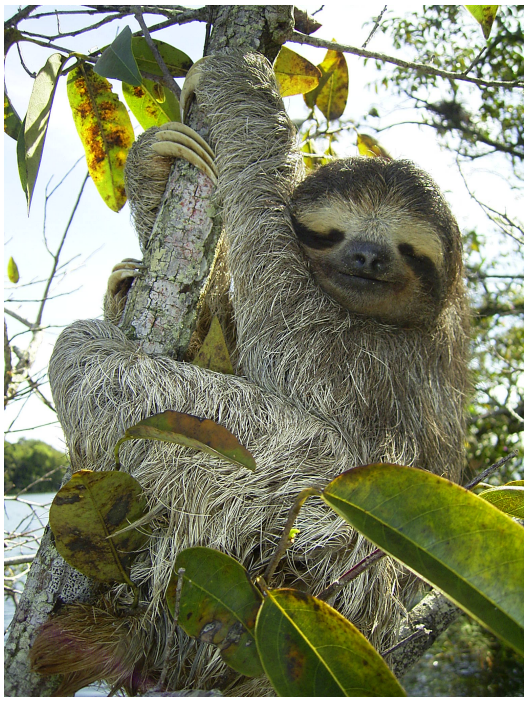


Figure 1: A three-toed sloth.

Moving slowly allows sloths to conserve energy and feed on a diet that provides very low nutritional value [42].

There is another natural benefit of moving slowly. Because many predators focus on movement as an indicator for possible prey, slow animals may draw less attention [20]. In fact, except for a few distinct species, most animals' camouflage is disrupted by movement [55]. This also applies to the predators themselves. In this case, camouflage is called "aggressive mimicry" [7]. The purpose of this is to keep prey unaware of the presence of the predator or to lure the prey into a trap. For technical applications this means that it can be beneficial to move slow when detection of the agents is not desirable.

Apart from evolutionary benefits in biological systems, there are good reasons to incorporate slowness in technical applications as well. If high velocities or accelerations need not be achievable, technical systems can be optimized for energy efficiency or low noise levels. An example for this are cars. Sports cars are optimized towards a high power output within the constraints of regulations on pollutant emissions. These types of cars generally have a worse fuel economy than compact cars that are optimized for efficiency, even when driven

under the same conditions.

Slow robots benefit from this fact in multiple ways. If the robot does not need to be able to accelerate fast, this reduces stress on mechanical parts of their powertrain. As a result, the wear of those parts is significantly reduced or the parts may be replaced with less strong lightweight parts. This reduces maintenance frequency and manufacturing costs of the robots. Especially for large swarms of robots, this is an important consideration. Building only as many fast and more expensive robots as necessary and having the rest of the team consist of cheap, slow robots is an intuitively reasonable strategy.

1.3 Multi-Robot Teams

Control and coordination of multi-robot teams have received significant attention in the last decade (see [9, 37, 48] and references therein). However, most research has dealt with homogeneous groups of robots, achieving an overall goal usually by working in the same way. Sometimes, homogeneity applies only to the hardware of the robots and the software is heterogeneous. In Dudek's taxonomy [16], robots that share the same hardware are classified as homogeneous, whereas robots that also have the same software are called identical. We will use the term heterogeneous for all sorts of teams with differences between the members. Despite receiving less attention than homogeneous teams, heterogeneity is an emerging topic in the field of distributed control.

Different types of heterogeneous teams have been investigated in the literature. In [46], heterogeneity in terms of skill sets is introduced through coevolution of the robot controllers to accomplish complicated tasks, while the used robots were homogeneous in terms of hardware. According to the taxonomy by Dudek [16], these robots are classified as homogeneous (same hardware), but not identical (same software).

Teams of robots with differing types of locomotion have been studied, especially the coordination of aerial and ground robots (e.g. [11, 17, 23, 52]). Other examples for heterogeneous locomotion include marsupial robots that can carry and deploy smaller robots with different capabilities. Trajectory and action planning for these kinds of teams are described in [14, 59]. Another example of this is presented in [49]. There, small and slow robotic

"scouts" are deployed in an area by a bigger, faster marsupial robot to perform surveillance tasks. A recent survey on the topic has been conducted in [26].

Sensory capabilities are one more source of heterogeneity. This has been applied to localization, mapping and exploration tasks. In [3], robots with high quality sensing units and robots with low quality sensing or no sensors at all are combined in a localization task. By sharing information about the measurements with a central computing unit, estimates for all robots' positions are computed. An example of modular robots with different equipped sensors is investigated in [22]. A sensor network consisting of a large number of low cost, low capability robots and a small number of guiding robots with highly capable sensors was built in [27]. Generally, smaller and cheaper robots with low sensing capabilities can be combined with a small number of more capable robots to accomplish tasks that require a distributed array of sensors and high sensing quality.

Other related research deals with the task assignment problem in heterogeneous computation systems [56]. It has been shown that heterogeneous multi-processor computer systems outperform homogeneous settings by up to 80% in extreme cases [4]. Again, the reduction of energy consumption is an issue that is dealt with, for instance in [50].

Despite these efforts, the topic of temporal heterogeneity is still lacking a systematic treatment. In this work, we are trying to narrow this gap. In fact, not only the solution of those problems with a given heterogeneous team will be the focus of this work, but also the optimal design of these teams.

We will now give a brief overview over the two fields in which we will apply heterogeneity. Both are instances of the assignment problem. For multi-robot systems and especially heterogeneous teams, one of the most important questions is "who does what?". It is essential to assign each robot's capabilities to a suitable task to maximize the utilization of their potential. In fact, this problem is one of the biggest differences between homogeneous and heterogeneous robotics. Where the task assignment plays a less vital role if every agent has the same capabilities, its importance increases with growing diversity. This issue is even more apparent in the following pursuit problem, where the wrong assignment can lead to failure.

1.4 Assignment Problems

Assignment problems are combinatorial optimization problems. The goal in an assignment problem is to find an assignment of workers to tasks that minimizes a linear cost function. Every worker has a specific cost associated with each task. The total cost of an assignment is the sum of the costs of all workers for their assigned tasks. Further, exactly one worker has to be assigned to every target and no worker can be assigned to multiple targets. Because the total cost of an assignment is the sum of each individual assignment's cost, we also refer to this problem as the linear assignment problem, as opposed to more complex non-linear assignment problems. The linear assignment problem is an instance of a linear program. It can also be described as the problem of finding a minimum weight independent edge set on a weighted bipartite graph.

An example given in [41] is the assignment of workers to jobs. Being varyingly efficient at different tasks, each worker has a performance rating assigned with each job. The assignment problem is now to find the assignment of workers to jobs that maximizes the total performance, i.e. the optimal utilization of resources under the constraint that every worker is assigned to exactly one job.

Being a linear program, the assignment problem can be solved with the simplex algorithm. However, there exist specialized algorithms for this problem that are much more efficient. Notable examples are the so-called Hungarian algorithm [41] and the decentralized auction algorithm [5].

The assignment problem is explained in more detail in Chapter 2. We will solve them for the case of robots traveling to target positions. The cost of an assignment will be determined by how quickly the task is accomplished and how energy-efficient the solution is. This is an instance of the general assignment of the heterogeneous agents to jobs and exemplifies the interpretation of velocities as skills that are applied to tasks.

In Chapter 3, the assignment problem is extended to moving targets. Special properties of the cost function that will be used in the pursuit allow us to efficiently solve the assignment in a decentralized manner. At the same time, the agents are also involved in a so-called pursuit game.

1.5 Pursuit Games

Game theory is used to model and analyze the behavior of interacting rational decision makers. The behaviors might be cooperation or conflict, depending on the game and the payoff functions for the players. The field of game theory dates back to 1928 when John von Neumann laid some of its foundations. It is now applied in numerous scientific directions including economics, political science and more recently control theory.

The games that we will look at are continuous time differential games, more precisely games of pursuit. These games include two rational entities: an evader and a pursuer. Different variants of pursuit games have been considered. An example is the "Lion and Man" problem. It consists of an agent (the lion) that tries to catch an evader (a man) within a circular arena. This problem has been extensively studied by Flynn [18, 19] and Lewin [35] among others. The questions that have been studied for various variants of this game include the existence of a value, conditions on the trajectories of man and lion and analytical solutions for the limit of the distance between the man and the lion when the lion is slower than the man.

The details of game theory are beyond the scope of this thesis. However, it is a useful tool for determining the worst-case scenario for an agent that pursues a target, namely an intelligent target that evades the agent in the best possible way. We will therefore use some game-theoretic formulations and results in Chapter 3.

1.6 Outline

The thesis consists of two main parts. In Chapter 2 we are investigating an assignment problem. In Section 2.1, an optimal assignment of mobile agents to target positions in terms of energy consumption and required time is introduced. The robotic team consists of robots of two different velocity levels: fast and slow. The results are extended to the assembly of optimal teams for the task in Section 2.2. In Section 2.3, we answer the question of what velocity levels are really optimal by developing a local optimization algorithm.

While Chapter 2 deals with the optimal solution to a transient problem, we consider a problem for large time-horizons in Chapter 3. We introduce the problem as a pursuit game

of multiple heterogeneous agents and multiple targets in Section 3.1. A measure for the cost of a target assignment is found in Section 3.2 and a decentralized algorithm for finding the optimal assignment of agents to targets is developed. We discuss the choice of the assignment cost in Section 3.3 for different scenarios. Again, an optimal team composition for the pursuit problem is determined in Section 3.4.

An implementation of all algorithms and practical results are described in Chapter 4. The necessary controllers for the nonlinear robot dynamics is introduced in Section 4.3. The assignment algorithm on the robotic testbed is described in Section 4.4. The implementation of the pursuit will be dealt with in Section 4.5

In Chapter 5, the work is summarized and conclusions are drawn. An outlook for future research directions is given in Section 5.1.

Chapter II

ASSIGNMENT

An assignment problem is a combinatorial optimization problem. It can be described as finding a minimum weight matching on a weighted bipartite graph. An interpretation is the assignment of workers to tasks, where each pair of worker and task has an assigned cost. In this case the problem is to find the assignment of workers to tasks that produces the minimum total cost while assigning exactly one worker per task and having each worker execute exactly one task.

In our case, the workers are robotic agents. The tasks they have to accomplish are target positions or rather driving to these target positions (pictured in Fig. 2). We can imagine a scenario where tasks need to be completed at the target positions (e.g. observing or interacting with something) and we want the robots to reach those targets as soon as possible, while also not wasting energy.

We introduce slowness here as a means to consume little energy by sacrificing some of the task execution speed. Depending on the target positions, it may be unnecessary to have a team consisting of only fast robots, and a few fast robots among a lot of slow robots could be sufficient. This question will be examined in Section 2.2, where we will assemble the

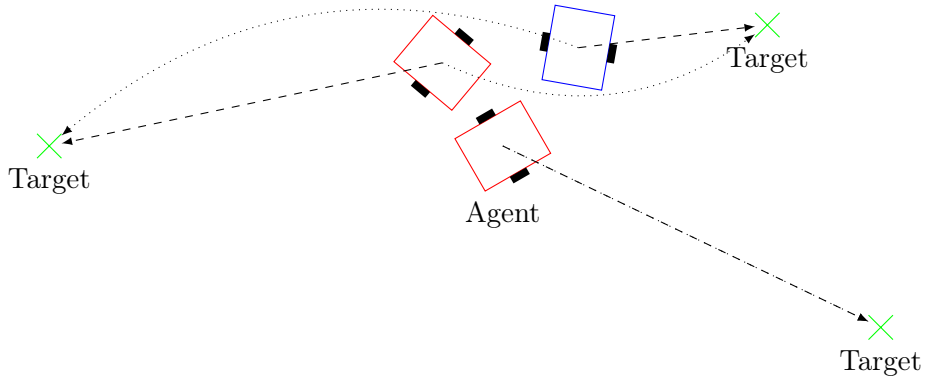


Figure 2: An example of an assignment problem. Two possible assignments are drawn with dotted and dashed arrows.

required team to optimally solve the assignment problem.

Because the agents are independent of each other, we consider the sum of the single assignment costs of all agents. This is called a linear assignment problem and is an example of a linear program. Solution methods for the linear assignment problem include the Hungarian algorithm [41] and the decentralized auction algorithm [5]. We will make use of the Hungarian algorithm to solve the assignment problem.

We will start this chapter by introducing a linear assignment problem for a team of agents.

2.1 A Linear Assignment Problem

Given a set of N agents with initial positions $x_i \in \mathbb{R}^2$ and N target positions $y_j \in \mathbb{R}^2$. We want the agents to travel to the target positions, using an optimal agent-target assignment. We define the set $\mathcal{N} = \{1, 2, \ldots, N\}$.

Definition The initial distance between an agent i and a target j is called $d_{i,j} = \|x_i - y_j\|$ with $i, j \in \mathcal{N}$. These values are assembled in a distance matrix $D = [d_{i,j}], i, j \in \mathcal{N}$. We define the instantaneous distance of agent i and target j to be $l_{i,j}(t)$.

In mobile robot scenarios, there are usually two important considerations. Naturally, the quickest possible completion of the task at hand is desired. However, most mobile robots have a limited supply of energy, which makes fast movement expensive. Therefore, one has to find a compromise between a fast completion of the task and energy conservation.

The cost function that will be used for the optimal assignment therefore incorporates energy expenditure and the distance of the agent to its target. The agent's distance to its target is used as a measure for the incompleteness of the task and is penalized. This is weighed against a cost for moving at high velocities to restrict energy consumption.

Definition The cost for assigning agent i to target j is defined as

$$c_{i,j} = \int_0^{T_{i,j}} \left[R v_i^2 + (1 - R)(\alpha + 1) l_{i,j}^{\alpha}(t) \right] \mathrm{d}t, \tag{1}$$

where $T_{i,j}$ is the time required by agent i to reach target j. $R \in [0, 1]$ and $\alpha \in \mathbb{N}_0$ are design parameters.

The parameter R can be used to adjust the relative importance of energy and distance for the cost, where $R = 1$ means that only energy is taken into consideration. Increasing α puts a higher weight on greater distances.

The goal is to find a bijection $P : \mathcal{N} \mapsto \mathcal{N}$ that maps every agent to its respective target. The total cost for an assignment P is the sum of the costs of the individual assigned agent-target pairs, formally

$$J(P) = \sum_{i \in \mathcal{N}} c_{i,P(i)} \ . \tag{2}$$

We will from now on use the simplified notation $d_i = d_{i,P(i)}$, $c_i = c_{i,P(i)}$ and $l_i = l_{i,P(i)}$.

Let $\mathcal{P}$ be the set of all possible assignments (permutations) P. We now want to find an optimal assignment P^* that satisfies

$$J(P^*) = J^* = \min_{P \in \mathcal{P}} J(P) \ . \tag{3}$$

This assignment is not guaranteed to be unique, so we are looking for any minimizer of J.

The instantaneous position of agent i is defined to be $p_i(t)$. Therefore, the current agent-target distance is expressed by $l_i(t) = \left\| p_i(t) - y_{P(i)} \right\|$. We assume the agent dynamics to be a simple integrator

$$\dot{p}_i = u, \quad p_i(0) = x_i, \tag{4}$$

where the dot represents a derivative w.r.t. time. We further use the control law

$$u = \begin{cases} v_i \frac{y_{P(i)} - p_i}{l_i(t)}, & p_i \neq y_{P(i)} \\ 0, & p_i = y_{P(i)} \end{cases} \tag{5}$$

to drive each agent directly towards its target with the constant velocity $v_i > 0$. This model assumes that each agent either stands still or moves at its maximal velocity v_i. Under the assumption that the velocity is directly controllable, the given control law is the fastest possible way for an agent to reach its target.

With this control law, we have

$$\dot{l}_i = \begin{cases} -v_i, & l_i \neq 0 \\ 0, & l_i = 0 \end{cases} \tag{6}$$

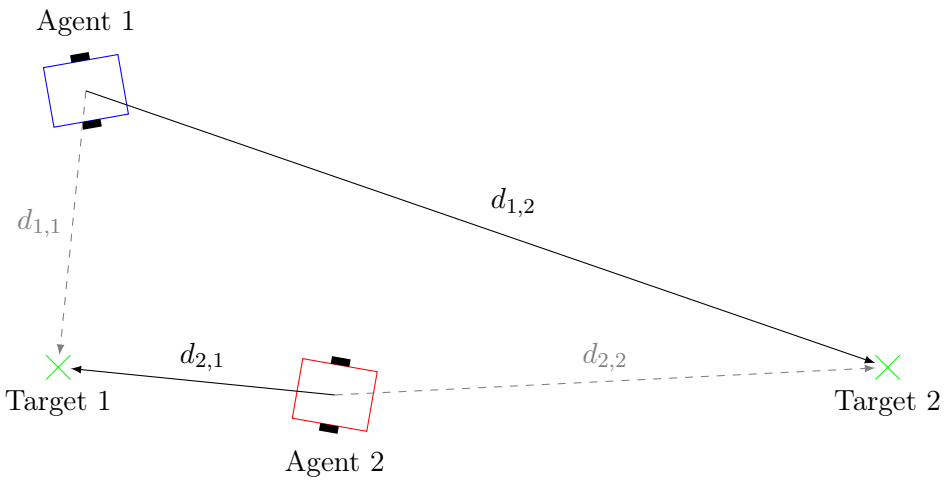


Figure 3: An example of the target assignment problem for two agents. Agent 1 is moving faster than agent 2.

which can be explicitly represented as

$$l_i(t) = d_i(1 - \frac{tv_i}{d_i}), \quad t \in [0, T_{i,P(i)}] \, . \tag{7}$$

With this, we can explicitly solve the integral in Eq. (1).

Lemma 2.1.1. *If agent i is assigned to target j and the control law in Eq. (5) is used for the system described in Eq. (4), we have the cost*

$$c_{i,j} = Rv_i d_{i,j} + (1 - R)\frac{d_{i,j}^{\alpha+1}}{v_i} \, . \tag{8}$$

To solve the linear assignment problem in Eq. (3), we assemble a cost matrix. It is defined as $C = [c_{i,j}]$ with $c_{i,j}$ from Eq. (8), $C \in \mathbb{R}_+^{N \times N}$. With this cost matrix, a suitable algorithm like the Hungarian algorithm [41] or a decentralized auction approach [39] can be used to solve the linear assignment problem.

Example 2.1.2. *Assume we have $N = 2$ agents and targets. They are configured as shown in Fig. 3. The distance matrix is*

$$D = \begin{bmatrix} 2.01 & 6.14 \\ 2.01 & 4.01 \end{bmatrix} \, . \tag{9}$$

Given the velocities $v_1 = 5$ and $v_2 = 1$ and setting $R = 0.3, \alpha = 2$, we have the cost function

$$c_{i,j} = 0.3 v_i d_{i,j} + 0.7 \frac{d_{i,j}^3}{v_i} \tag{10}$$

and can assemble the cost matrix

$$C = \begin{bmatrix} 4.15 & 41.53 \\ 6.29 & 46.17 \end{bmatrix} . \tag{11}$$

We can see that the optimal assignment is $P(1) = 2$ *and* $P(2) = 1$. *This assignment does not minimize the traveled distance. It is drawn in Fig. 3 as solid black arrows.*

2.2 Velocity Distribution

So far the agents' velocities v_i were assumed to be given. Instead we will now view them as a design variable for the agent group. We use this way to introduce heterogeneity into the system. In particular, let the velocity of each agent be selectable between a high velocity v_f and a low velocity v_s. The number of different velocity levels is arbitrary, but we want to represent the notion of slow and fast movement that has been outlined earlier.

We define the velocity choice function $Q : \mathcal{N} \times \mathcal{N} \mapsto \{v_s, v_f\}$ and set $v_i = Q(i, P(i))$ for an assignment of agent i to target $P(i)$. Since the agent-target cost c_i is a function of the agent's velocity, we will denote it as $c_i(v_i)$ or $c_i\left(Q(i, P(i))\right)$. The problem of finding the best speed distribution can be described as

$$(P^*, Q^*) = \underset{P,Q}{\operatorname{argmin}} \sum_{i \in \mathcal{N}} c_i\left(Q(i, P(i))\right) . \tag{12}$$

The optimal velocity of an agent is independent of the other agents' assignments. In fact, the optimal $Q^*(i, j)$ for given v_s, v_f depends only on $d_{i,j}$. For this reason, the optimal velocity $Q^*(i, j)$ for $i, j \in \mathcal{N}$ can be determined a priori for every element $c_{i,j}$ of the cost matrix C.

Lemma 2.2.1. *For* $\alpha \geq 1$ *there is a critical distance*

$$\Delta = \left(\frac{R}{1-R} v_s v_f\right)^{\frac{1}{\alpha}} \tag{13}$$

at which the costs for slow and fast movement are equal, i.e. $c_{i,j}(v_s) = c_{i,j}(v_f)$ *if* $d_{i,j} = \Delta$. *The optimal velocity is*

$$Q^*(i, j) = \begin{cases} v_f, & d_{i,j} > \Delta, \\ v_s, & d_{i,j} \leq \Delta. \end{cases} \tag{14}$$

Proof. The difference between the costs for fast and slow movement at a non-zero distance d between agent and target is

$$c(v_f) - c(v_s) = Rd(v_f - v_s) + (1-R)d^{\alpha+1}(\frac{1}{v_f} - \frac{1}{v_s}) \; . \tag{15}$$

We show that this is positive for $d < \Delta$ and negative for $d > \Delta$, i.e. fast movement yields the lower cost for great distances and higher cost for distances lower than Δ. Let $d = \sigma\Delta$. Because we assumed $d > 0$, we can also divide the difference by d without changing the sign of the term:

$$R(v_f - v_s) + (1-R)(\sigma\Delta)^{\alpha}(\frac{1}{v_f} - \frac{1}{v_s}) \tag{16}$$

$$= R(v_f - v_s) + (1-R)\sigma^{\alpha}\frac{R}{1-R}v_s v_f(\frac{1}{v_f} - \frac{1}{v_s}) \tag{17}$$

$$= R(v_f - v_s) - \sigma^{\alpha}R(v_f - v_s) = (1-\sigma^{\alpha})R(v_f - v_s) \; . \tag{18}$$

Since $R > 0$ and $v_f > v_s$, this term is negative for $\sigma > 1$, i.e. $d > \Delta$, and positive for $\sigma < 1$. For $\sigma = 1$ we have $d = \Delta$ and the difference between the costs for slow and fast movement is zero. □

Since slow and fast movement are equally expensive at $d_{i,j} = \Delta$, the choice to have $Q^*(i,j) = v_s$ for $d_{i,j} = \Delta$ was arbitrary. The agent-target cost as a function of the distance is displayed in Fig. 4 for two different speed levels. The curves intersect at the critial distance Δ.

Remark If we set the parameter α in the agent-target cost from Eq. (8) to zero, we have

$$c_{i,j}^{\alpha=0}(v_i) = Rd_{i,j}v_i + (1-R)\,\frac{d_{i,j}}{v_i} = d_{i,j}\left(Rv_i + \frac{1-R}{v_i}\right) \; . \tag{19}$$

In this case, the optimal choice of v_i is independent of the agent-target distance $d_{i,j}$. This leads to a homogeneous team.

According to Lemma 2.2.1 the optimal velocity for every agent-target pair can be determined a priori. In fact, the optimal velocity of agent i only depends on its own assignment

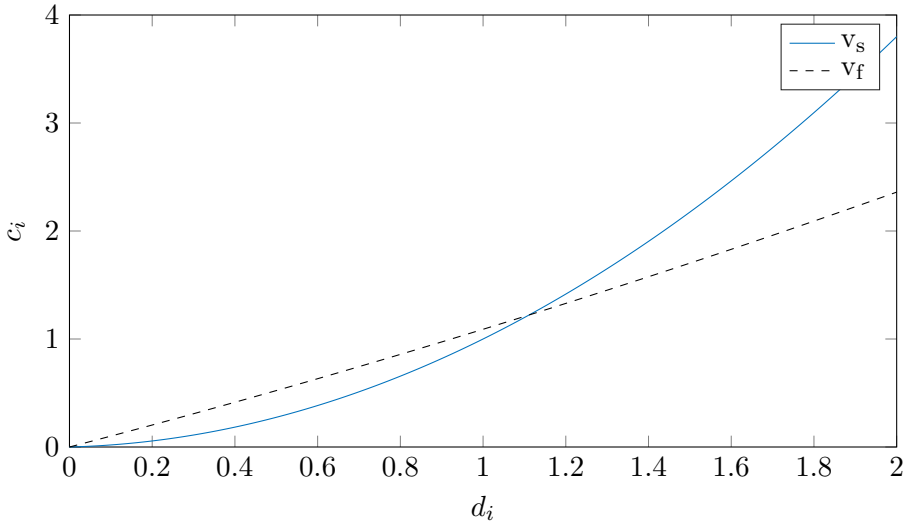


Figure 4: The cost c_i as a function of d_i for two different agent velocities.

$P(i)$. In particular, it is independent of the velocity or the assignment of all the other agents. This is why, given P, we have

$$\min_{Q} \sum_{i \in \mathcal{N}} c_i \left(Q(i, P(i)) \right) = \sum_{i \in \mathcal{N}} \min_{Q} c_i \left(Q(i, P(i)) \right) . \tag{20}$$

This leads us to the following statement.

Theorem 2.2.2. *The optimization problem in Eq.* (12) *is equivalent to*

$$P^* = \operatorname*{argmin}_{P \in \mathcal{P}} \sum_{i \in \mathcal{N}} c_i \left(Q^*(i, P(i)) \right) , \tag{21}$$

where $Q^*(i, P(i))$ is the optimal choice of v_i according to Eq. (14)*.*

This is now a linear assignment problem again and can be solved by suitable algorithms. We do now know how to simultaneously assign agents to targets and assemble the optimal heterogeneous team for two given velocity levels.

Example 2.2.3. *We are revisiting Example 2.1.2, using the same initial configuration and parameters. We are now allowing to pick the velocities of the agents to be either $v_s = 1$ or $v_f = 5$. The critical distance for $R = 0.3$ is*

$$\Delta = \frac{0.3}{1 - 0.3} \cdot 5 = 2.14 . \tag{22}$$

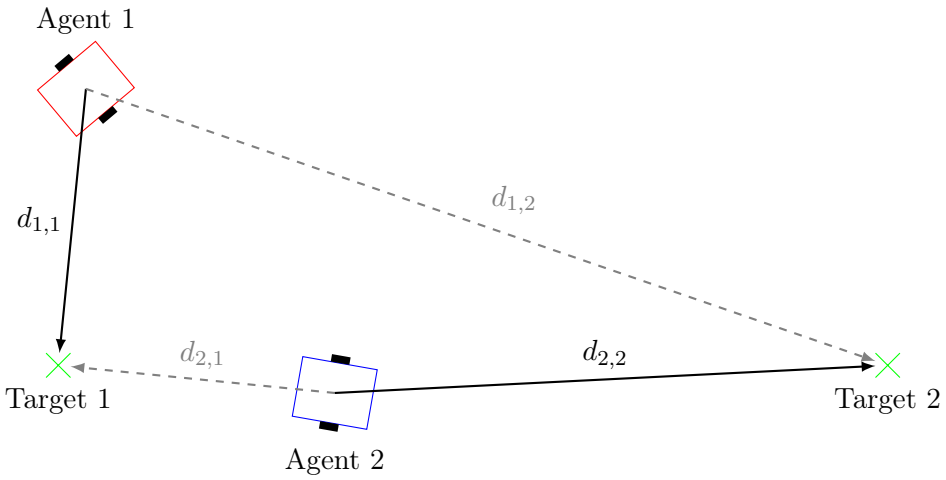


Figure 5: The result of the team assembly problem in Example 2.2.3. Agent 1 is moving slow and agent 2 is moving fast.

We have $d_{1,1} < \Delta$ *and* $d_{2,1} < \Delta$ *and therefore* v_s *is the preferred velocity for those agent-target pairs. On the other side,* $d_{1,2}$ *and* $d_{2,2}$ *are greater than* Δ *and the optimal velocity would be* v_f*. The cost matrix using the optimal velocities for every agent-target pair is*

$$C = \begin{bmatrix} 4.15 & 41.53 \\ 4.15 & 15.00 \end{bmatrix} . \tag{23}$$

The optimal assignment is $P(1) = 1$ *and* $P(2) = 2$ *where* $v_1 = v_s$ *and* $v_2 = v_f$*. This is the opposite configuration to the one from the earlier example. The result is illustrated in Fig. 5. The optimal assignment is plotted as solid black lines.*

2.3 Choice of Velocity Levels

We are interested in assembling the optimal homogeneous team for the given task of moving to target positions. We described how one can pick the agents' velocities from two given values. To extend the problem formulation further, we may ask which values we should actually choose for the heterogeneous movement velocities v_s and v_f. In the team design process, this could be viewed as the design of the actual robots in terms of the required movement velocities. This extends the problem from Eq. (12) to

$$(P^*, Q^*, v_s^*, v_f^*) = \underset{P,Q,v_s,v_f}{\operatorname{argmin}} \sum_i c_i \left(Q(i, P(i), v_s, v_f)\right) . \tag{24}$$

We now denote $Q(i, P(i), v_s, v_f)$ also as a function of the variable velocities v_s and v_f.

Given a pair of velocities v_s and v_f, we can find the optimal Q^* and P^* as described in the section before. However, the optimal values of the velocities v_s^* and v_f^* in turn depend on the assignment P. To find the optimal velocities given an assignment P, we need to find a solution to

$$(v_s^*, v_f^*) = \underset{v_s, v_f}{\operatorname{argmin}} \sum_{i \in \mathcal{N}} c_i \left(Q^*(i, P(i), v_s, v_f) \right) . \tag{25}$$

We will first describe the solution of this problem for a homogeneous team. It will be extended to multiple homogeneous groups forming a heterogeneous team.

Lemma 2.3.1. *Choosing an assignment P is equivalent to creating a set of the assigned agent-target distances. The homogeneous version of Eq.* (25) *for a set of distances $\mathcal{D}$ is given by*

$$v^* = \underset{v \in \mathbb{R}_+}{\operatorname{argmin}} \sum_{d \in \mathcal{D}} \left[Rvd + (1 - R) \frac{d^{\alpha+1}}{v} \right] . \tag{26}$$

It is solved by the optimal homogeneous velocity

$$v^* = \sqrt{\frac{1 - R}{R} \frac{\sum_{d \in \mathcal{D}} d^{\alpha+1}}{\sum_{d \in \mathcal{D}} d}} . \tag{27}$$

Proof. We are looking for a minimum of the cost

$$J_{\mathcal{D}}(v) = \sum_{d \in \mathcal{D}} \left[Rvd + (1 - R) \frac{d^{\alpha+1}}{v} \right] . \tag{28}$$

For $v > 0$ this is a smooth function of the velocity. It is further convex, because it is a weighted sum of the convex functions v and $\frac{1}{v}$. If we choose R from the open interval $(0, 1)$, the function $J_{\mathcal{D}}$ indeed has a minimum at some positive velocity. The minimizer of the cost satisfies

$$\left. \frac{\partial J_{\mathcal{D}}}{\partial v} \right|_{v = v^*} = 0 . \tag{29}$$

The solution in Eq. (27) is obtained by setting the derivative of $J_{\mathcal{D}}$ w.r.t. v to zero and solving for v. The derivative of $J_{\mathcal{D}}$ is

$$\frac{\partial J_{\mathcal{D}}}{\partial v} = \sum_{d \in \mathcal{D}} \left[Rd - (1 - R) \frac{d^{\alpha+1}}{v^2} \right] . \tag{30}$$

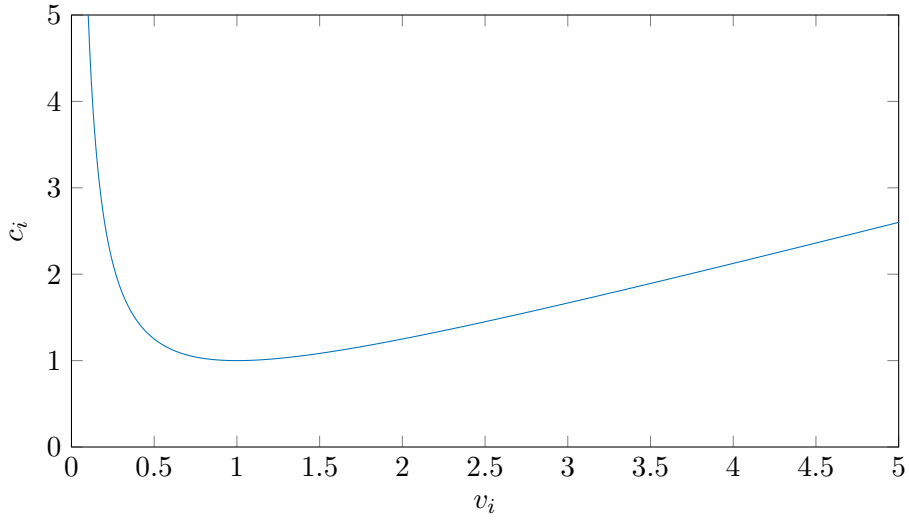


Figure 6: The cost c_i as a function of v_i for $R = 0.5$ and $d_i = 1$.

So with Eq. (29) and by rearranging we have

$$(v^*)^2 = \frac{1-R}{R} \frac{\sum_{d \in \mathcal{D}} d^{\alpha+1}}{\sum_{d \in \mathcal{D}} d} \,. \tag{31}$$

Because the velocity has to be positive, we obtain the original statement. □

An example of the cost $c_{i,j}$ as a function of the agents' velocity is displayed in Fig. 6.

Lemma 2.3.1 only applies to homogeneous groups. We will therefore split up the heterogeneous into two homogeneous groups. One group consists of the agents moving at the high velocity v_f and the other of agents moving at v_s. We can then compute the optimal values for those velocities for every possible division of the team.

Definition Let $\mathcal{D}_P = \left\{ d_{i,P(i)}, i \in \mathcal{N} \right\}$ be the set of agent-target distances in an assignment P. Then $\mathcal{D}_f^\delta = \{\, d \in \mathcal{D}_P \mid d > \delta \,\}$ is the set of distances greater than some $\delta \in \mathbb{R}_+$ and $\mathcal{D}_s^\delta = \{\, d \in \mathcal{D}_P \mid d \leq \delta \,\}$ is the set of distances that are less than δ.

For every $\delta \in \mathcal{D}_P$ we get a division of $\mathcal{D}_P$ into $\mathcal{D}_f^\delta$ and $\mathcal{D}_s^\delta$. Let $\mathcal{D}_f^\delta$ be the set of distances for the fast moving agents. Then according to Lemma 2.3.1, we have the optimal high velocity

$$v_f^\delta = \sqrt{\frac{1-R}{R} \frac{\sum_{d \in \mathcal{D}_f^\delta} d^{\alpha+1}}{\sum_{d \in \mathcal{D}_f^\delta} d}} \tag{32}$$

and the corresponding result for slow movement v_s^δ.

Remark There are 2^{N-1} different ways to split up the set of distances into two subsets. However, according to Lemma 2.2.1 the optimal division of the set is at a critical distance Δ. Because this critical distance is a function of the not yet known velocity levels, we have to compare all cases where $\mathcal{D}_P$ is split at some threshold δ.

From Lemma 2.3.1 we can also determine bounds on the velocity levels v_s and v_f.

Corollary 2.3.2. *The optimal movement velocities* (v_s^*, v_f^*) *are bounded by*

$$\sqrt{\frac{1-R}{R}}(\min \mathcal{D}_P)^{\alpha/2} \leq v_s^*, v_f^* \leq \sqrt{\frac{1-R}{R}}(\max \mathcal{D}_P)^{\alpha/2} \,. \tag{33}$$

Proof. We will prove the statement only for the lower bound on v_s. The upper bound and v_f can be proved analogously. We can bound a sum of positive elements in a set $\mathcal{D}$ from below by its minimal element as follows:

$$(\min \mathcal{D})^{\alpha} \sum_{d \in \mathcal{D}} d \leq \sum_{d \in \mathcal{D}} d^{\alpha+1} \,. \tag{34}$$

Further, we have $\min \mathcal{D}_P \leq \min \mathcal{D}_s^{\delta}$ for any δ because $\mathcal{D}_s^{\delta}$ is a subset of $\mathcal{D}_P$. It follows that

$$(\min \mathcal{D}_P)^{\alpha} \sum_{d \in \mathcal{D}_s^{\delta}} d \leq \sum_{d \in \mathcal{D}_s^{\delta}} d^{\alpha+1} \,. \tag{35}$$

or after division by the (positive) sum:

$$(\min \mathcal{D}_P)^{\alpha} \leq \frac{\sum_{d \in \mathcal{D}_s^{\delta}} d^{\alpha+1}}{\sum_{d \in \mathcal{D}_s^{\delta}} d} \,. \tag{36}$$

This holds for any (non-empty) subset $\mathcal{D}_s^{\delta}$ of $\mathcal{D}_P$. If we plug this into Eq. (27) we obtain the original statement. □

Since we can find the optimal velocities for any division, we are interested in the division that yields the lowest cost

$$J = \min_{\delta \in \mathcal{D}_P} \sum_{i \in \mathcal{N}} c_i \left(Q^*(i, P(i), v_s^{\delta}, v_f^{\delta}) \right) \,. \tag{37}$$

Since there are only N different divisions of $\mathcal{D}_P$, simple computation and comparison of all values is sufficient. Let δ^* be the minimizer of Eq. (37).

Lemma 2.3.3. *The resulting values* $(v_s^{\delta^*}, v_f^{\delta^*})$ *are the solution* (v_s^*, v_f^*) *to Eq.* (25).

As remarked earlier, the solution to the problem has to be one of the divisions $(\mathcal{D}_s^\delta, \mathcal{D}_f^\delta)$ for $\delta \in \mathcal{D}_P$. Because we can compute the optimal values for the velocities (v_s, v_f) for any division, we have indeed exhausted all plausible solutions.

Remark The critical distance Δ is in general not equal to the the threshold δ. In addition, the division of $\mathcal{D}_P$ into the sets $\mathcal{D}_f^\delta$ and $\mathcal{D}_s^\delta$ is not guaranteed to match the slow/fast-distribution given by Δ. We call this case an invalid division.

Lemma 2.3.4. *The optimal division by the threshold δ^* is always valid in the sense that δ^* and $\Delta(v_s^{\delta^*}, v_f^{\delta^*})$ divide $\mathcal{D}$ into the same subsets, i.e.*

$$\mathcal{D}_s^\Delta = \mathcal{D}_s^{\delta^*} \ . \tag{38}$$

Proof. Assume δ^* is the threshold distance such that $(v_s^{\delta^*}, v_f^{\delta^*})$ yield the lowest cost for all divisions of $\mathcal{D}_P$. Let this cost be c^{δ^*}. If the division is invalid, this means there is a Δ such that

$$\mathcal{D}_s^\Delta \neq \mathcal{D}_s^{\delta^*} \ . \tag{39}$$

For any set of velocities (v_s, v_f), the critical distance Δ yields a division that minimizes every agent's cost. If the divided sets are different, this means that we have $c^\Delta \leq c^{\delta^*}$. Further, there is a $\tilde{\delta} \in \mathcal{D}_P$ that divides the set in the same way as Δ:

$$\mathcal{D}_s^\Delta = \mathcal{D}_s^{\tilde{\delta}} \ . \tag{40}$$

That means that $c^{\tilde{\delta}} < c^{\delta^*}$. This contradicts the assumption that δ^* is the optimal threshold.

□

We can now find the optimal velocities given an assignment and the optimal assignment given the velocities. Therefore, we approach the problem in Eq. (24) with an iterative algorithm.

Algorithm 2.3.5. *Let P^k be the solution of the assignment problem in the k-th iteration and $(v_f, v_s)^k$ be the solution of Eq.* (25)*.*

1. *Find the optimal movement velocities $(v_f, v_s)^k$ for the assignment P^k.*

2. *Using* $(v_f, v_s)^k$*, determine the solution* P^{k+1} *to the assignment problem.*

3. *Repeat until* $P^{k+1} = P^k$*.*

We arbitrarily initialize the assignment P^0 *as* $P^0(i) = i, i \in \mathcal{N}$*.*

To analyze the result of this algorithm, we look at the optimal cost for any given velocity levels. Recall the total cost J of an assignment from Eq. (2). Because it is a function of both the assignment and the velocities, we will now denote it as $J(P, v_s, v_f)$.

Definition We introduce the cost $J_{P^*} : \mathbb{R}_+^2 \mapsto \mathbb{R}_+$. It is defined as

$$J_{P^*}(v_s, v_f) = \min_{P \in \mathcal{P}} J(P, v_s, v_f) = \min_{P \in \mathcal{P}} \sum_{i \in \mathcal{N}} c_{i,P(i)}(Q^*(i, P(i), v_s, v_f)) \; . \tag{41}$$

The function returns the best achievable cost given the velocity levels v_s and v_f. In other words, the value of J_{P^*} is the current cost after the second step of Algorithm 2.3.5 for any (v_s, v_f).

Lemma 2.3.6. *A global minimizer of* $J_{P^*}(v_s, v_f)$ *is a solution to Eq.* (24)*.*

Proof. This statement follows from the fact that J_{P^*} returns the lowest achievable cost (through choice of P) for any (v_s, v_f). As stated in Lemma 2.2.1, the optimal Q directly follows from (P, v_s, v_f). A global minimum of this function in (v_s, v_f) is therefore also a minimum in P and Q. □

For specific configurations, the optimal assignment for a given set of velocities is not unique. For this reason, we impose the following assumption.

Assumption 2.3.7. *For every* $\tilde{P} \in \mathcal{P}$*, we have a minimal point* (v_s^*, v_f^*) *of* $J(\tilde{P}, v_s, v_f)$*. We assume* $\tilde{P}$ *to either not be the best assignment and therefore*

$$J(\tilde{P}, v_s^*, v_f^*) > J_{P^*}(v_s^*, v_f^*) \tag{42}$$

or, if it is the best assignment and

$$J(\tilde{P}, v_s^*, v_f^*) = J_{P^*}(v_s^*, v_f^*) \; , \tag{43}$$

we assume it to be the unique solution of the assignment problem at this point.

Theorem 2.3.8. *If Assumption 2.3.7 is satisfied, Algorithm 2.3.5 reaches a local minimum of the function J_{P^*} after a finite number of iterations.*

Proof. The first step of the algorithm always finds the global minimum of $J(P^k, v_s, v_f)$ w.r.t. (v_s, v_f). We have exactly two cases to consider.

If there exists another assignment $\hat{P}$ with a lower cost than P^k at this point, the solution to the assignment problem is not $P^{k+1} = P^k$ and therefore the algorithm continues.

If every other assignment yields a strictly greater cost, we have found the optimal assignment and

$$J(P^k, v_s^k, v_f^k) = J_{P^*}(v_s^k, v_f^k) \ . \tag{44}$$

Because we assumed the optimal point to be unique in Assumption 2.3.7, the solution to the assignment problem is once again $P^{k+1} = P^k$ and the algorithm terminates. Because $J(\tilde{P}, v_s, v_f)$ is a continuous function of (v_s, v_f) for any assignment $\tilde{P} \in \mathcal{P}$, Eq. (44) holds in a neighborhood of (v_s^k, v_f^k) and P^k is indeed a local minimum of J_{P^*}. □

Remark Assumption 2.3.7 is not a strong assumption, because it is only violated for a small number of initial conditions.

Remark If Assumption 2.3.7 is violated, a simple extension of the algorithm can guarantee convergence. If the solution of the assignment problem is found to be non-unique, we can compute the optimal velocities for all solutions of the assignment problem and continue the algorithm at the best of these points.

2.4 Example: A Sensor Coverage Problem

A possible application for this sort of assignment problem can be found in another domain of networked systems. Sensor coverage problems are concerned with the distribution of sensing nodes in a domain with the goal of maximizing overall sensor coverage quality. These problems and a decentralized algorithm for their solution are described in [12]. By introducing a Voronoi partitioning (i.e. assigning every point in the domain to its closest agent, see Fig. 7) and then using a variant of Lloyd's algorithm as the control law for the agents, locally optimal coverage of the domain can be achieved. If the goal is not uniform

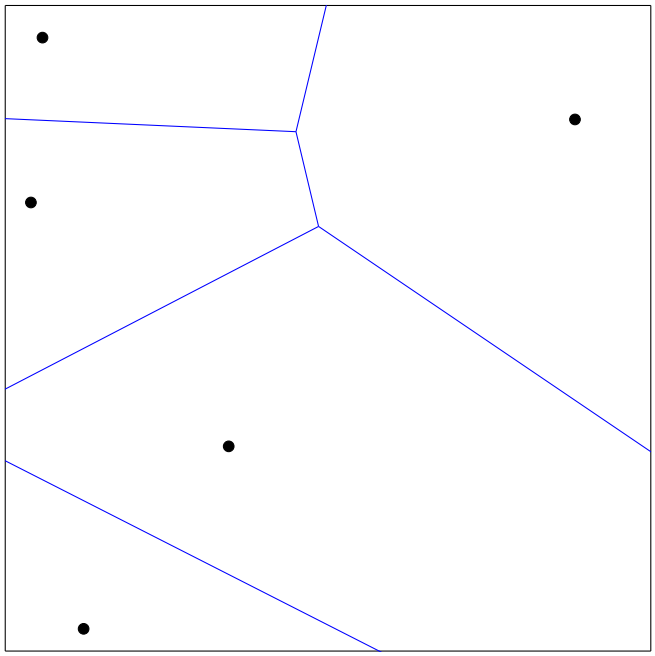


Figure 7: A Voronoi partitioning for 5 nodes.

coverage of the whole domain, a density function can be used to achieve a denser coverage in more important areas.

In [32], the methods are extended to deal with time-varying density functions. These density functions may be generated by an operator as a human machine interface as proposed in [15] or they might be created otherwise to make the network adapt to changing requirements.

As an example, one can imagine a sensor network to acoustically observe an area. When sounds are detected at some location, a higher coverage at this point would be desirable to find the cause of the recorded noise. To achieve this, the density function for the coverage problem would be increased at this location. Unfortunately, the algorithms described in [12] and [32], while not generally unsuited, are not designed for heterogeneous teams.

In the described event, the reasonable action would be to send a fast member of the team to the new area of interest to investigate. However, the known coverage control algorithms can only achieve this behavior if the area of interest is contained in the Voronoi cell of a fast agent (i.e. if the closest agent to this point of interest is a fast agent). Otherwise, the slow agents will slowly creep towards the new point of interest. This is partly due to the

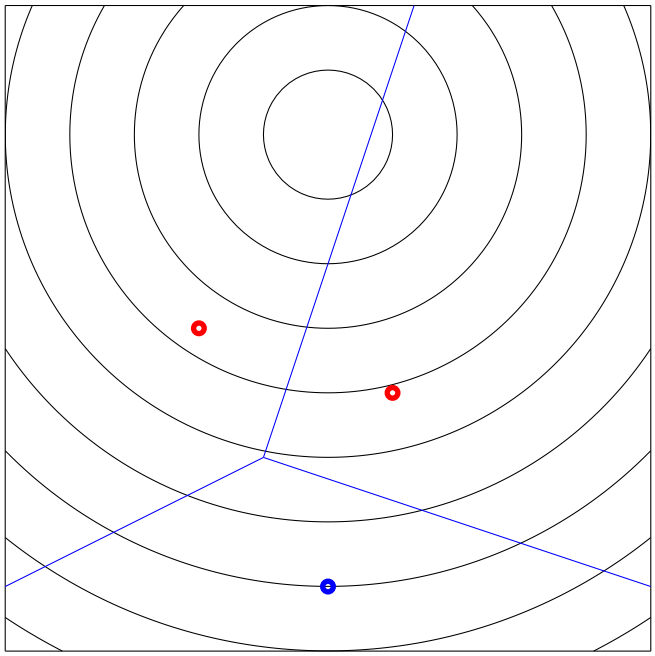


Figure 8: An example of the coverage problem where the fast agent on the bottom has no knowledge about the maximum in the coverage density on the top. The circles are a contour plot of the coverage density. The blue lines are the edges of the Voronoi partitioning.

decentralized nature of the algorithms, where information about non-adjacent cells in the Voronoi partitioning is not available to the agents. An example where this would happen is depicted in Fig. 8. Informally speaking, the two other agents above the agent on the bottom obstruct its view on the maximum in the coverage density.

If we give up decentralization, all information is available to a centralized decision maker. A solution now is to turn the coverage problem into an assignment problem. The centralized decision maker simulates Lloyd's algorithm until it converges and then assigns agents to the resulting target positions in an optimal way. An example of this can be seen in Fig. 9. If we directly compare the coverage cost (see [12]) of the assignment algorithm to the standard Lloyd algorithm (with agents moving at their maximal velocities), scenarios like the one described can lead to a much faster decrease. This is visible in Fig. 10.

2.5 Simulation

An simulative example of the whole process including finding the optimal velocity levels, assembling the best team and solving the assignment problem can be seen in Fig. 11. The

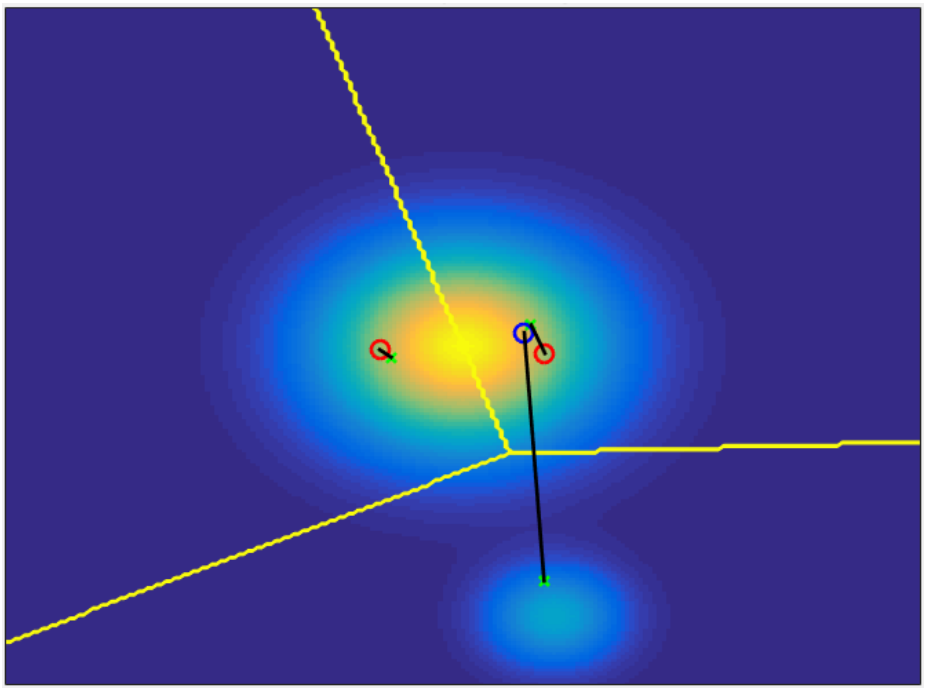


Figure 9: An example of the assignment algorithm in a coverage problem. The fast agent (blue circle) is assigned to the new target on the bottom. The crosses mark the target positions after convergence of the Lloyd algorithm, the yellow lines are the resulting Voronoi partitioning.

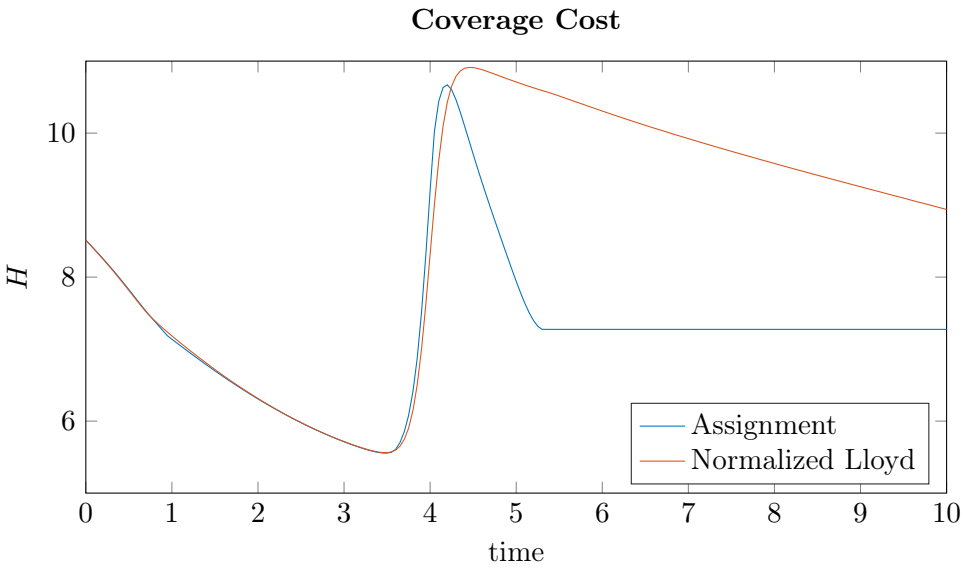
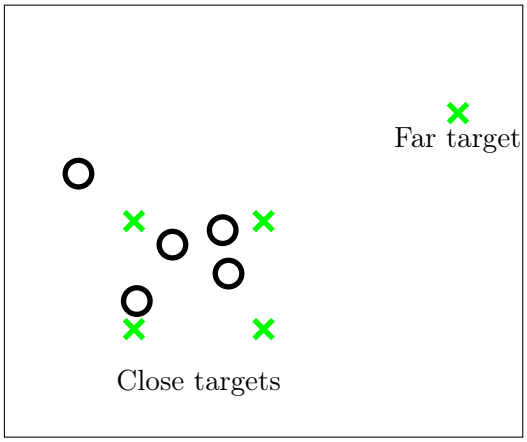
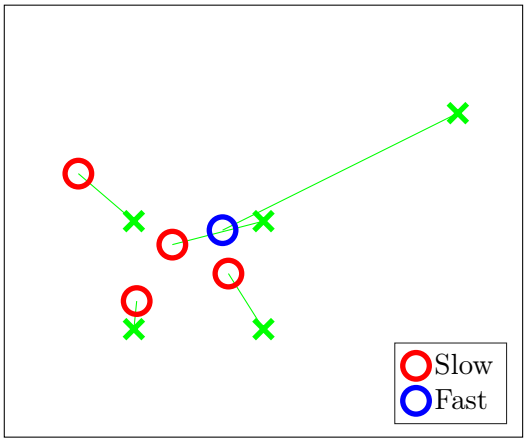


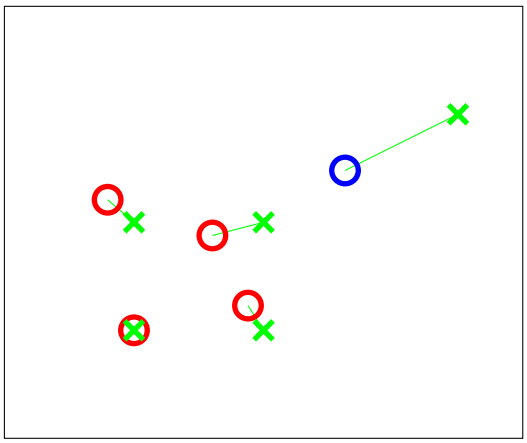
Figure 10: A comparison of the coverage cost H for the regular Lloyd algorithm (with normalized velocities) and the assignment algorithm. Lower values correspond to better coverage of the area.



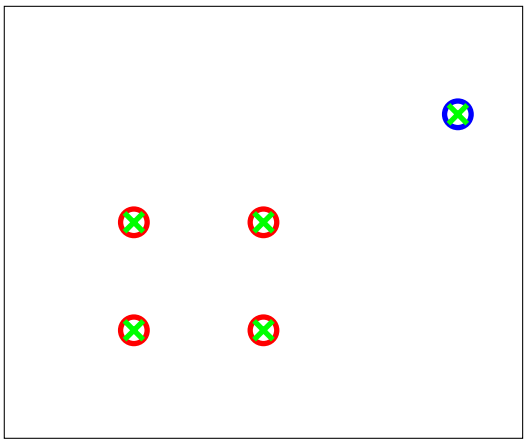
(a) Target assignment and team composition problem setup. The circles represent agents, targets are marked with an x.



(b) After finding the optimal velocity levels and team composition. One of the agents should move fast, because its target is far away.



(c) After letting the agents move for some time.



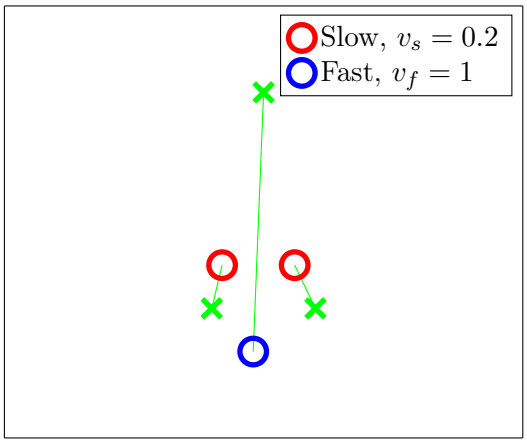
(d) The final positions of the agents on the targets.

Figure 11: A simulative example of finding optimal velocity levels, the team composition and a target assignment.

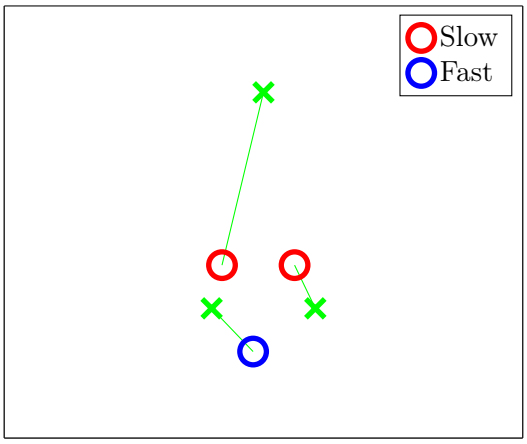
solution supports our intuitive guess that one agent which needs to cross a higher distance to the far target will have to move at the high velocity. To conserve energy, the agents that are assigned to close targets are chosen to move slowly.

Figure 12 shows two results for different values of the energy weighing coefficient R. Again, the result confirms our intuition. If a quick execution has the higher priority (low values of R, Fig. 12a), the fast agent is assigned to the target that is the furthest away, despite increasing the total traveled distance and specifially the distance traveled at high velocity. If the value of R is high (right side, Fig. 12b), the emphasis is on energy consumption. This leads to an assignment where agents need much more time to complete the task, but the energy consumption is very low due to the smaller distance to be traveled.

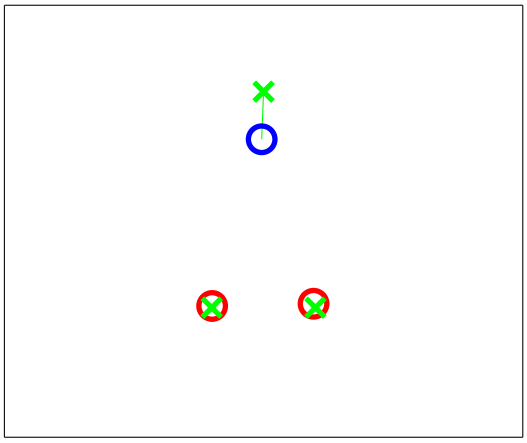
It should be noted that different values of R do not lead to different assignments in every scenario. In many cases, the assignment that is energy-optimal is also the one that leads to the quickest task completion. Different results for different values can generally be observed when there is a single target that is far away from the group of robots and the rest of the target.



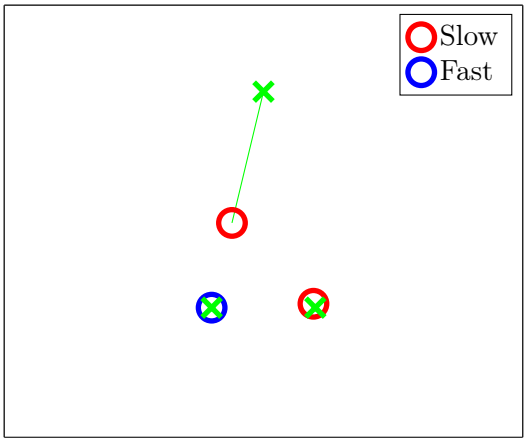
(a) Assignment problem solution for $R = 0.2$. Reaching the targets quickly has a higher priority.



(b) Assignment problem solution for $R = 0.8$. Energy saving is valued more than a quick task completion.



(c) The left assignment at a time t_1.



(d) The right assignment at t_1. The task completion takes more time.

Figure 12: A simulative example of assignment for two different weights R for $\alpha = 1$. A lower value of R (seen on the left) puts a higher emphasis on a quick task completion, high values lead to energy-optimal assignments. All parameters except R are identical in the left and right examples.

Chapter III

PURSUIT

After studying the assignment of a temporally heterogeneous set of agents to stationary targets in Chapter 2, we seek to extend the problem to one with a moving set of targets. In this chapter, agents are supposed to reach or at least come close to a set of moving targets. Because targets are moving, capture is not always guaranteed. The teams we consider in this chapter further consist of more different agents. Instead of two distinct velocities at which agents can move, we will now use a number of different maximum velocities. Each agent therefore has some maximum movement speed, but can also move slower than that.

Because we do not know the behavior of the targets beforehand and we will use some stochastic elements within our algorithms, we will look at the qualitative behavior of the agents and targets for large time horizons. The fundamental questions then are if capture (i.e. the positions of agent and target coincide) occurs and how close an agent may come to a target if capture is not possible. We call this problem target pursuit (see Fig. 13). In this problem we assume the worst possible case for the agents, namely targets that actively evade the targets in the best possible way. This is an instance of a differential game, more precisely a game of pursuit and evasion. We will make use of a few game-theoretic concepts and formulations later in this chapter.

A possible real-world scenario for this theoretical problem is a multi-agent surveillance task: A heterogeneous group of agents is supposed to guard an area. When intruders enter this area, it is the agents' task to stay close to those intruders and observe them. If malicious behavior is detected, further units could be deployed to take action against the intruder. If an agent is too slow for its assigned surveillance target, the distance between the two increases and the observation quality decreases.

For a group of agents, the problem is twofold: in the small scale, each agent needs to pursue its assigned target and stay as close as possible. This problem is largely dependent

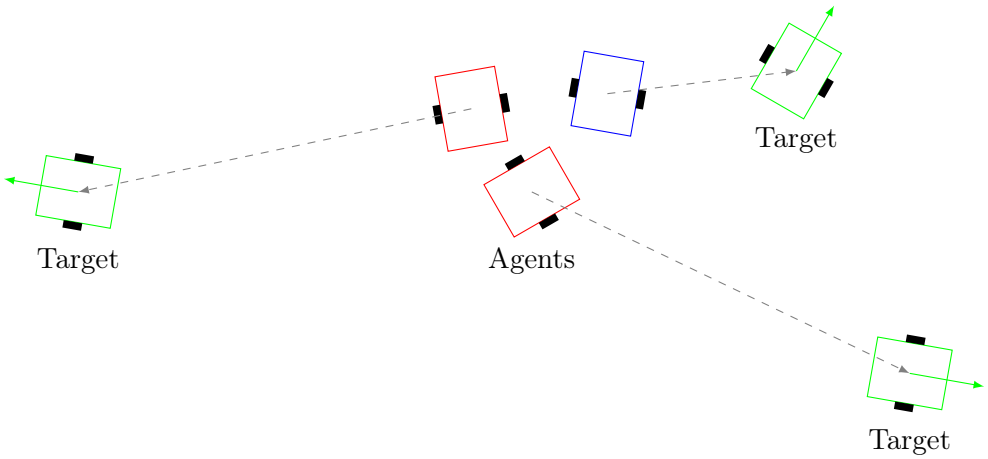


Figure 13: An example of the team pursuit problem. The moving targets could be another set of robots.

on the actual dynamical model of the agents and targets and will be dealt with in more detail in Chapter 4. On the greater scale, the agents need to find an assignment of agents to targets such that the overall surveillance task can be accomplished by all agents. We will introduce a decentralized algorithm for this purpose.

For simplicity, we will assume that the number of targets matches the number of agents and every agent is assigned to exactly one target. If there are less targets than agents, virtual targets with zero velocity can be introduced. If the number of targets exceeds the number of agents, we can look at a subset of the targets of the same number as the set of agents. However, we will not go into this case in this work.

We will once again consider the optimal team composition for the pursuit task in Section 3.2. First, we need to introduce the problem and the strategy employed by the team of agents. We start the chapter by introducing a linear, single-integrator target pursuit in two dimensions.

3.1 The Linear Target Pursuit

The linear target pursuit, also known as the "Lion and Man" game, is the most simple version of the pursuit problem. We are now looking at a single member of the robotic team, assigned to a single target. Both agent and target have positions x and y in the plane $\mathbb{R}^2$. The movement of agent and target may be unbounded or constrained to stay within an

arena. The task of the agent is to reach the target position.

With the two-dimensional velocity input $u(t) \in \mathbb{R}^2$, the dynamics of the agent are

$$\dot{x} = u, \quad x(0) = x_0 \in \mathbb{R}^2 \tag{45}$$

and the norm of the input is saturated by

$$\|u\| \leq \hat{u} \, . \tag{46}$$

Further, we have a moving target position. We will think about it as another agent that can move in the plane in the same way as the pursuer. We assume the target dynamics to be the same as the agent's:

$$\dot{y} = v, \quad y(0) = y_0 \, . \tag{47}$$

Its input is saturated by

$$\|v\| \leq \hat{v} \, . \tag{48}$$

The saturation on the input values of the agent's and target's dynamics are upper bounds on the movement speed of the agent and the target. While this model does not describe the actual robot dynamics that are introduced in Chapter 4, it serves as a simplification and is a good enough approximation.

The physical distance between the agent and the target is defined as $d(t) = \|x(t) - y(t)\|$. In a pursuit game, the evader seeks to maximize this distance while the pursuer tries to minimize it.

We are now considering the question whether the agent can drive the distance to the target to zero for the cases that it is slower, faster or equally fast as the target. It has been shown that the intuitive solution is correct for the cases that the agent is slower or faster than the target.

If $\hat{u} < \hat{v}$, the agent is slower than the target. Therefore, the target can evade the agent forever. If agent and target are confined to a circular arena, the agent can reach and maintain a constant distance to the target. This will be discussed later in this chapter.

If the agent can move faster than the target, we have $\hat{u} > \hat{v}$ and the agent can reach the target in finite time.

In the case that $\hat{u} = \hat{v}$, the success of the pursuit depends on the bounds on the state. If we assume an infinitely large arena, the evading target can keep the distance to the agent constant forever. If the positions are restricted to a circular arena however, the agent can get arbitrarily close to the target, even though the target can still avoid capture. In this case, we have

$$\lim_{t \to \infty} d(t) = 0 \ , \tag{49}$$

but $d(t) > 0$ for all $t > 0$. This was shown by Besicovitch (see [36]).

In the following when looking at teams of agents, the position of agent i will be denoted as x_i and its maximum velocity as $\hat{u}_i$. The variables x and $\hat{u}$ denote the vectors of all agents' positions and velocities respectively. The same holds for the targets. The vector y_j is the position of target j and $\hat{v}_j$ is its maximum velocity.

3.2 Pursuit Target Assignment

Given a set of N agents and N targets. Each agent and each target has an identifying integer index i. The set of indices is denoted as $\mathcal{N} = \{1, 2, ..., N\}$. As introduced before, x_i is the position of agent i and $\hat{u}_i$ describes the maximum velocity of that agent. Let the indices of the agents be ordered ascending by the maximum velocity $\hat{u}_i$. In the single-integrator case, agent 1 is the slowest and agent N is the fastest agent. Equally, the targets are sorted by their maximum velocity $\hat{v}_j$. The order of the agents ensures

$$\hat{u}_1 < \hat{u}_2 < ... < \hat{u}_N \tag{50}$$

and similarly for the targets

$$\hat{v}_1 < \hat{v}_2 < ... < \hat{v}_N \ . \tag{51}$$

We assume that no two agents or two targets have identical velocities. This ensures that there is a unique solution to the assignment problem. If there are duplicates, any permutation of the identical agents or targets yields an identical solution.

The goal is to find a bijection $P : \mathcal{N} \mapsto \mathcal{N}$ that maps every agent to its respective target. The assignment P can be viewed as a permutation of the numbers from 1 to N. The set $\mathcal{P}$ is

the set of all possible permutations. Further, let P^{-1} denote the inverse of the assignment, i.e. the mapping from a target to the assigned agent.

The discrepancy between the velocities of an agent and its assigned target is

$$\delta(i,j) = |\hat{u}_i - \hat{v}_j| \ . \tag{52}$$

We consider a cost function $\Delta : \mathcal{P} \mapsto \mathbb{R}$. It is the sum of the squared discrepancies

$$\Delta(P) = \sum_{i \in \mathcal{N}} \delta^2(i, P(i)) \ . \tag{53}$$

This cost is minimized if the target velocities match the pursuing agent velocities as closely as possible. Differences in the speed of an agent and its target are penalized. This is intuitively the right choice when the target is faster than the agent. The penalty for an agent being unnecessarily fast is chosen to achieve a more even distribution of velocity differences in the case that the agents are faster than the targets. It does not change the limit of the agent-target distances for large t.

We now introduce an algorithm that minimizes Δ in a decentralized fashion. It is similar to swap type sorting algorithms in that it swaps two agents' targets if this decreases the cost function. To determine whether a swap reduces Δ, only local information is required.

A version of the algorithm is as follows:

Algorithm 3.2.1. *In every iteration, pick a random pair of agents (i,j). Then swap the targets of these agents if*

$$\delta^2(i, P(j)) + \delta^2(j, P(i)) < \delta^2(i, P(i)) + \delta^2(j, P(j)) \ . \tag{54}$$

The criterion for performing a swap ensures that the cost Δ always decreases when targets are swapped.

Lemma 3.2.2. *When Eq. (54) is satisfied, swapping the targets of the agent pair (i,j) decreases the cost Δ.*

Proof. Without loss of generality, we are assuming $i < j$. The correctness of this statement can be seen when taking apart the cost Δ:

$$\Delta(P) = \sum_{k=1}^{i-1} \delta^2(k, P(k)) + \delta^2(i, P(i)) + \sum_{l=i+1}^{j-1} \delta^2(l, P(l)) + \delta^2(j, P(j)) + \sum_{m=j+1}^{N} \delta^2(m, P(m)) \ . \tag{55}$$

The cost of the assignment P' where the targets of agent i and j are flipped is

$$\Delta(P') = \sum_{k=1}^{i-1} \delta^2(k, P(k)) + \delta^2(i, P(j)) + \sum_{l=i+1}^{j-1} \delta^2(l, P(l)) + \delta^2(j, P(i)) + \sum_{m=j+1}^{N} \delta^2(m, P(m)) \ . \tag{56}$$

We can see that

$$\Delta(P) - \Delta(P') = \delta^2(i, P(i)) + \delta^2(j, P(j)) - (\delta^2(i, P(j)) + \delta^2(j, P(i))) \tag{57}$$

and because of Eq. (54), this difference is positive. Therefore, the cost of P' is lower than that of P. □

In general, this algorithm is not guaranteed to solve the linear assignment problem because there may be assignments that cannot be improved by a single swap of two targets. It is however suitable for this specific problem, as is shown in the following.

Lemma 3.2.3. *For every assignment other than $P^* : P^*(i) = i, i \in \mathcal{N}$ there is a swapping pair (i, j) that decreases the cost Δ.*

Proof. Assume $P(1) \neq 1$. Then swapping the targets of agent 1 and $P^{-1}(1)$ decreases the cost Δ. Let $P(i) = 1$ and $P(1) = j$ for some $i, j \neq 1$. The ordering of the elements ensures that $\hat{u}_1 < \hat{u}_i$ for $i = 2, 3, ...$ and $\hat{v}_1 < \hat{v}_j$ for $j = 2, 3, ...$ Therefore, we have

$$\hat{u}_1(\hat{v}_j - \hat{v}_1) < \hat{u}_i(\hat{v}_j - \hat{v}_1) \ . \tag{58}$$

We can use this to show the original claim by performing some elementary transformations. We have

$$\begin{aligned}
\hat{u}_1\hat{v}_j - \hat{u}_1\hat{v}_1 &< \hat{u}_i\hat{v}_j - \hat{u}_i\hat{v}_1 \ , \\
-\hat{u}_1\hat{v}_j - \hat{u}_i\hat{v}_1 &> -\hat{u}_i\hat{v}_j - \hat{u}_1\hat{v}_1 \ , \\
\hat{u}_1^2 - 2\hat{u}_1\hat{v}_j + \hat{v}_j^2 + \hat{u}_i^2 - 2\hat{u}_i\hat{v}_1 + \hat{v}_1^2 &> \hat{u}_i^2 - 2\hat{u}_i\hat{v}_j + \hat{v}_j^2 + \hat{u}_1^2 - 2\hat{u}_1\hat{v}_1 + \hat{v}_1^2 \ , \\
(\hat{u}_1 - \hat{v}_j)^2 + (\hat{u}_i - \hat{v}_1)^2 &> (\hat{u}_i - \hat{v}_j)^2 + (\hat{u}_1 - \hat{v}_1)^2 \ , \\
\delta^2(1, j) + \delta^2(i, 1) &> \delta^2(i, j) + \delta^2(1, 1) \ .
\end{aligned}$$

Hence, the criterion from Eq. (54) is satisfied and swapping the targets of the agent pair $(1, P^{-1}(1))$ decreases the cost.

If $P(1) = 1$, the same reasoning holds for swapping target 2 to agent 2 in the reduced set of agents $\mathcal{N} \setminus \{1\}$ and so forth. Therefore, the claim holds for all $P \in \mathcal{P} \setminus \{P^*\}$. □

Corollary 3.2.4. *The assignment $P^* : P^*(i) = i, i \in \mathcal{N}$ is a unique minimizer of the cost Δ.*

Proof. Because the set $\mathcal{P}$ is finite, the minimum of $\Delta(P)$ for $P \in \mathcal{P}$ exists and has to be attained for some $\tilde{P} \in \mathcal{P}$. Since Lemma 3.2.3 states that for any $P' \in \mathcal{P} \setminus \{P^*\}$ there is another assignment $P'' \in \mathcal{P}$ such that $\Delta(P'') < \Delta(P')$, none of the assignments in $\mathcal{P} \setminus \{P^*\}$ can be this minimizer $\tilde{P}$. That only leaves the element P^* and therefore $\tilde{P} = P^*$ is the minimizer of Δ. □

Theorem 3.2.5. *Algorithm 3.2.1 reaches the optimal assignment P^*.*

Proof. The algorithm does not visit any permutation twice, because it strictly decreases Δ when the assignment is changed. According to Lemma 3.2.3, the algorithm can always decrease Δ in a single step unless the optimal assignment is reached. Choosing a random pair of agents in every iteration ensures that with probability one a swap will be performed at some point if there is a swap that decreases Δ. Since the set of all permutations $\mathcal{P}$ is finite, the optimal assignment P^* is attained at some point. □

3.2.1 Decentralized Deterministic Assignment

The algorithm can be decentralized because the decision to swap targets is based solely on local information. The communication network between the agents can be described as an undirected graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of the nodes (agents) and $\mathcal{E}$ contains all edges between the nodes. If an edge (i, j) is contained in $\mathcal{E}$, then nodes i and j can communicate. We call i and j neighbors. The set of all neighbors of node i is denoted as $\mathcal{N}_i$.

To determine whether or not a swap would be beneficial, an agent needs to know its partner's velocity and the velocity of its partner's target. This directly follows from Eq. (54).

To account for the restricted communication, Algorithm 3.2.1 is rephrased to a decentralized version on a communication graph.

Figure 14: An example of an OLG, $\hat{u}_1 < \hat{u}_2 < \hat{u}_3 < \hat{u}_4$.

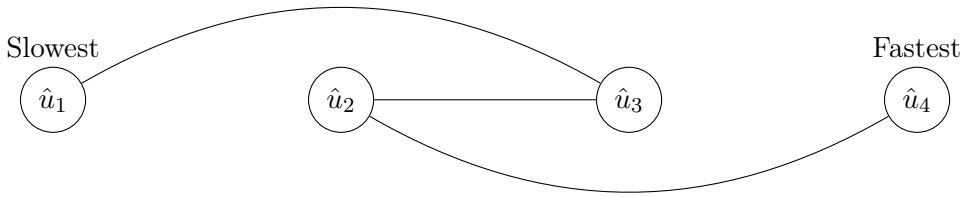


Figure 15: An example of a line graph that is not an OLG.

Algorithm 3.2.6. *In every iteration, an agent i chooses one of its graph neighbors $j \in \mathcal{N}_i$ at random. If the swapping condition in Eq.* (54) *is satisfied, agents i and j swap their targets.*

Instead of choosing a random neighbor, the agent could also communicate with all of its neighbors and choose the swap partner such as to maximize the decrease of Δ. We will assume that not more than one agent tries to swap targets with a specific agent at the same time. This can be interpreted as a centralized entity that selects one agent randomly at every time step. This agent can then swap targets with one of its graph neighbors or do nothing.

Obviously, this algorithm does not reach an optimal assignment for some initial conditions if the graph G is not connected. Therefore, we want to find requirements on the graph that ensure successful assignment for any initial conditions.

Definition We call a graph an Ordered Line Graph (OLG) if it is a line graph and the order of the nodes on the line corresponds to the order of the agents by the norm of their velocity saturation $\hat{u}_i$, as introduced earlier.

An example of an OLG is displayed in Fig. 14. The OLG is unique for a set of agents when none of them have the same velocities. A line graph that is not an OLG is displayed in Fig. 15. The nodes on the line do not have the right order corresponding to their velocities.

If Algorithm 3.2.6 reaches the optimal assignment P^* for any initial assignment P_0 for a given graph, this graph will be called admissible. We will call the class of all such admissible graphs $\mathcal{G}_a$. Conversely, if a graph is inadmissible, there is at least one initial assignment that in no case leads to the optimal assignment by using Algorithm 3.2.6. The condition "in no case" is added to encase all possible sequences of random agent selections.

Lemma 3.2.7. *An OLG is admissible.*

Proof. The optimal assignment P^* is a sorted version of the initial assignment. In an ordered line graph, all possible swaps will be between graph neigbors, i.e. between an agent i and its neighbor $i+1$. From the order of agents we know that $\hat{u}_i < \hat{u}_{i+1}$ for $i \in \{1, 2, ..., N-1\}$. Further assume that $k < l$ and therefore $\hat{v}_k < \hat{v}_l$. We have

$$\hat{u}_i(\hat{v}_l - \hat{v}_k) < \hat{u}_{i+1}(\hat{v}_l - \hat{v}_k) \ . \tag{59}$$

From the proof of Lemma 3.2.3 we know that

$$\delta^2(i,k) + \delta^2(i+1,l) < \delta^2(i,l) + \delta^2(i+1,k) \ . \tag{60}$$

This means that when comparing neighboring nodes, the target with the lower velocity is swapped to the agent with the lower velocity.

In fact, this can be viewed as a decentralized variant of the well-known bubble sort algorithm, where the agents correspond to the indices of an array and the target velocities are the numbers that need to be sorted. For a sufficient number of swaps between neighbors, this algorithm has been shown to yield the correct result [31]. □

Lemma 3.2.8. *If edges are added to an admissible graph $G \in \mathcal{G}_a$, the resulting graph is still admissible. This means that every spanning supergraph of an admissible graph is admissible.*

Proof. Adding edges to the communication graph only adds new possible swaps, but doesn't take away any. We already showed that in an admissible graph there is always an available swap that decreases the cost in Lemma 3.2.3. This existence is naturally not broken by adding new possible swaps. □

Corollary 3.2.9. *Any spanning subgraph of an inadmissible graph is inadmissible.*

Proof. The statement is proved by contradiction. Assume there was an inadmissible graph $G_i = (\mathcal{V}, \mathcal{E}_i)$ and an admissible spanning subgraph $G_s = (\mathcal{V}, \mathcal{E}_s)$ with $\mathcal{E}_s \subset \mathcal{E}_i$. But since Lemma 3.2.8 states that any spanning supergraph of the admissible graph G_s is admissible, G_i has to be admissible. This contradicts the initial assumption. □

Lemma 3.2.10. *Any graph G that contains an OLG as a spanning subgraph is admissible.*

Proof. This statement follows directly from Lemma 3.2.7 and Lemma 3.2.8. □

We now have established that the set of all supergraphs of an OLG is a subset of all admissible graphs $\mathcal{G}_a$. It remains to show that the two sets are equal, i.e. that there are no admissible graphs that do not have an OLG as a spanning subgraph.

Lemma 3.2.11. *Any graph that does not contain an OLG as a spanning subgraph is not admissible.*

Proof. To prove this claim, we will show that a complete graph is not admissible anymore if one of the edges of the OLG is removed. We will then infer by Corollary 3.2.9 that any graph that is missing one of these edges is inadmissible.

To show that a graph is not admissible, it suffices to find an assignment $P^{\#}$ that is not optimal and that can not be improved by a single step of Algorithm 3.2.6.

Let us consider a complete graph $K_N = (\mathcal{V}, \mathcal{E}_K)$. Now we remove a single edge that lies on the OLG. Let this edge be $(i, i+1)$ for $i \in \{1, 2, ..., N-1\}$. We get the set of edges $\mathcal{E}'_i = \mathcal{E}_K \setminus \{(i, i+1)\}$ and the reduced graph $G'_i = (\mathcal{V}, \mathcal{E}'_i)$.

Now assume an almost ordered assignment. We have $P^{\#}(j) = j$ for $j \in \mathcal{N} \setminus \{i, i+1\}$ and $P^{\#}(i) = i+1$, $P^{\#}(i+1) = i$. All targets are assigned to the right target except for i and $i+1$ which are swapped.

The only action that would reduce the cost Δ is swapping these two targets in the right order. Every other possible swap would swap a slow target that is assigned to a slow agent and a fast target that is assigned to a fast agent, reversing this order and thereby increasing the cost. Since the nodes i and $i+1$ are not graph neighbors on G'_i, they cannot communicate with each other and this swap is not possible. Therefore, no cost-reducing swap

can be executed and the sub-optimal assignment $P^{\#}$ is the terminal assignment. Hence, the graphs G'_i are not admissible.

Every graph that does not contain an OLG as a spanning subgraph is itself a spanning subgraph of one of these reduced complete graphs. With Corollary 3.2.9, we can conclude that every graph without an OLG as a spanning subgraph is inadmissible. □

We have shown the sufficiency and necessity of an OLG as a spanning subgraph of the agent communication graph. Lemma 3.2.10 and Lemma 3.2.11 can be combined to the following.

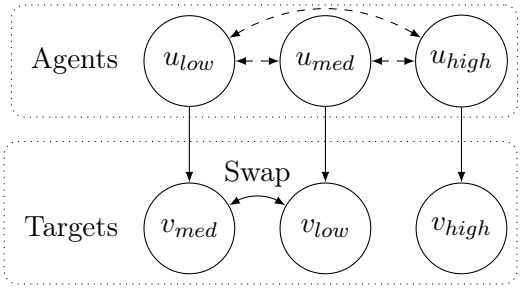
Theorem 3.2.12. *Algorithm 3.2.6 reaches the optimal assignment P^* for any initial assignment P_0 on a graph G if and only if G contains a spanning subgraph that is an OLG.*

Example 3.2.13. *We are considering the two assignments in Fig. 16. From the proof of Lemma 3.2.3 we know that for a pair of agents, assigning the slower target to the slower agent yields lower cost. We see that the only swap that would reduce the cost in the example would be between the left and the center agent. In the left picture (Fig. 16a), this swap will eventually happen and the optimal assignment will be reached.*

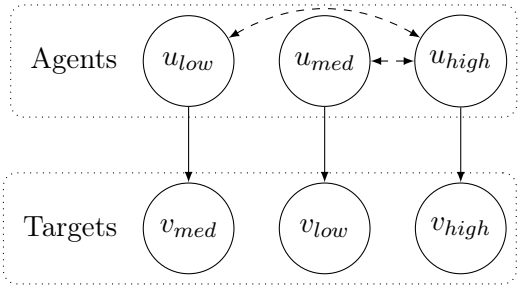
In the right example (Fig. 16b) however, this swap is not allowed because the left two agents can not communicate with each other. The optimal assignment could still be reached by swapping the slow target to the fast agent first and then to the slow agent. However, this swap (or any allowed swap) would in fact increase the cost and will therefore not be performed. This means that the optimal assignment can not be reached in the right example without increasing the cost in between. The agents are stuck in a local minimum of the cost function.

3.2.2 Stochastic Assignment

Because the ordered line graph condition on the graph stated in Theorem 3.2.12 is rather restricting and presumptively hard to verify in a decentralized fashion, we seek an algorithm that is more robust to different graph structures. By adding randomness to the algorithm,



(a) An assignment of agents to targets. The indicated swap produces the optimal assignment.



(b) The same assignment with a reduced communication graph. The swapping algorithm will not achieve the optimal assignment, because the left two agents cannot communicate with each other.

Figure 16: Two different communication graphs for the same sub-optimal assignment. Depending on the graph, the optimal assignment can or cannot be reached by the swapping algorithm.

local minima of the cost function Δ should be escaped. This is common practice in global optimization algorithms such as simulated annealing [51].

3.2.2.1 Potential Game Formulation

We will formulate the problem as a potential game and address it by using the Metropolis-Hastings-Algorithm. This will enable us to relax the necessary and sufficient condition on the graph to a simple demand for connectivity, albeit at the cost of leaving the optimal assignment occasionally. This method has been proposed to the author by Dr. Daniel Pickem and is highly influenced by [44].

The following notation will be used.

Definition The tuple $G = (\mathcal{N}, \mathcal{A}, \{U_i(.)\}_{i\in\mathcal{N}}, \{\mathcal{R}_e\}_{e\in\mathcal{E}}, \Phi(.))$ is called a constrained potential game, where

- $\mathcal{N}$ is the set of N players
- $\mathcal{A} = \mathcal{A}_1 \times ... \times \mathcal{A}_N$ is the product set of the agents' action sets $\mathcal{A}_i$
- $U_i : \mathcal{A} \mapsto \mathbb{R}$ are the agents' local potentials
- $\mathcal{R}_e : \mathcal{A} \mapsto \mathcal{A}_i \times \mathcal{A}_j$ is the restricted bilateral action set for a pair of agents $e = \{i, j\} \in \mathcal{E}$, further called edge

- $\Phi : \mathcal{A} \mapsto \mathbb{R}$ is the global potential of a joint action

The local potential of an edge $e = \{i, j\}$ is defined for a joint action a as the sum of the local potentials of its agents $U_e(a) = U_i(a) + U_j(a)$. Further, the local and global potentials are aligned if for all edges $e \in \mathcal{E}$, all bilateral actions $a_e, a'_e \in \mathcal{R}_e$ and actions of other agents $a_{-e} \in \prod_{k \in \mathcal{N} \setminus e} \mathcal{A}_k$ we have

$$U_e(a'_e, a_{-e}) - U_e(a_e, a_{-e}) = \Phi(a'_e, a_{-e}) - \Phi(a_e, a_{-e}) \ . \tag{61}$$

In the assignment problem, no agent can change its action unilaterally because this would lead to invalid assignments. Instead, agents always can only swap targets to maintain a bijective assignment. This is why we define the notion of bilateral actions a_e on edges of the graph $e \in \mathcal{E}$. The restricted action set $\mathcal{R}_e$ is a function that maps a joint action a to the set of permissible actions of an edge. The restricted action set always contains two bilateral actions: swapping the actions of the two agents or keeping the current joint action. For an edge $e = \{i, j\}$ this is expressed as

$$\mathcal{R}_e = \{(a_i, a_j), (a_j, a_i)\} \ . \tag{62}$$

It can be seen that a joint action $a \in \mathcal{A}$ corresponds to an assignment $P \in \mathcal{P}$. In fact, the complete action set $\mathcal{A}$ also contains invalid assignments where more than one agent is assigned to a target. Therefore, we call the set of all admissible actions $\mathcal{A}_P$.

For the local potential function, we will use the squared velocity mismatch $U_i(a) = \delta^2(i, P(i)) = \delta^2(i, a_i)$. The global potential is the sum of all agents' local potentials $\Phi(a) = \sum_{i \in \mathcal{N}} U_i(a)$ and is equal to the total velocity mismatch $\Delta(P)$. It can be seen that the local and global potential are indeed aligned and satisfy Eq. (61).

The local potential U_i of an agent can be computed with local information about the agent and its assigned target only. Similarly, the information required to compute the local potential of a pair of agents is the union of the two agents' information.

3.2.2.2 Markov Chain Monte Carlo Sampling

The potential game can be described as a Markov chain. The states are all possible joint actions (assignments), transitions between states are target swaps of two agents. An example

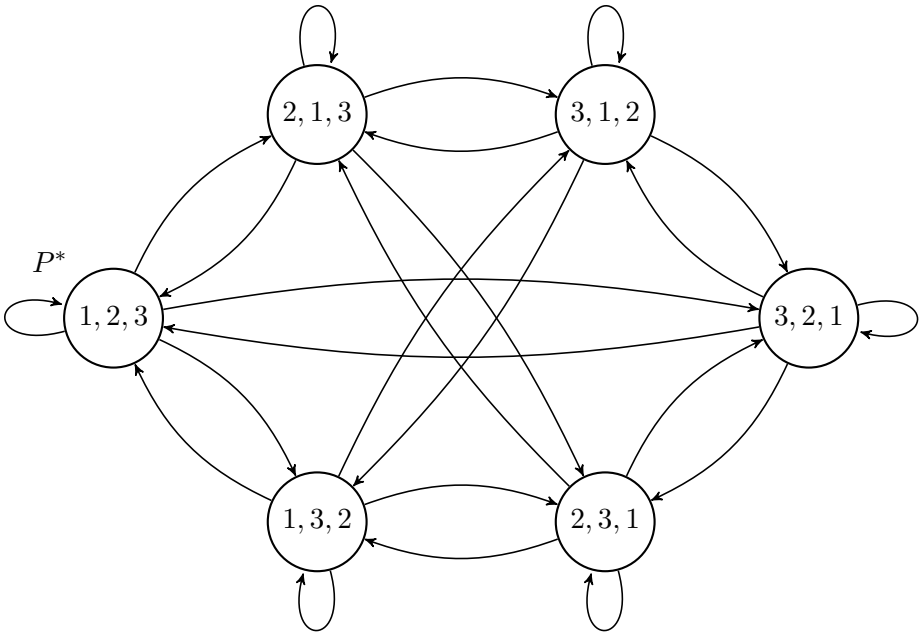


Figure 17: Markov chain of all permutations (assignments) P for 3 agents on a complete graph.

of this Markov chain for three agents on a complete graph is presented in Fig. 17. Another example of a Markov chain is presented in Fig. 18. The underlying communication graph does not contain an OLG. The sub-optimal state $P^{\#}$ is absorbing when the deterministic algorithm is used. Using the Metropolis-Hastings-algorithm, we will now design the transition probabilities in such a way that the optimal action is the stochastically stable state.

We will keep the notation a for the joint actions and therefore for the states of the Markov chain.

Definition We define a Markov chain with the following properties:

- There exists a unique stationary distribution of the states.
- The probability of being in state a at a given time is $\pi(a)$.
- When the system is in state a, the probability of transitioning to state a' is $Pr[a \to a']$.

We want the optimal joint action a^* (corresponds to the optimal assignment P^*) to be the stochastically stable state of the system (defined in Section 3.2.2.2). To achieve this, we

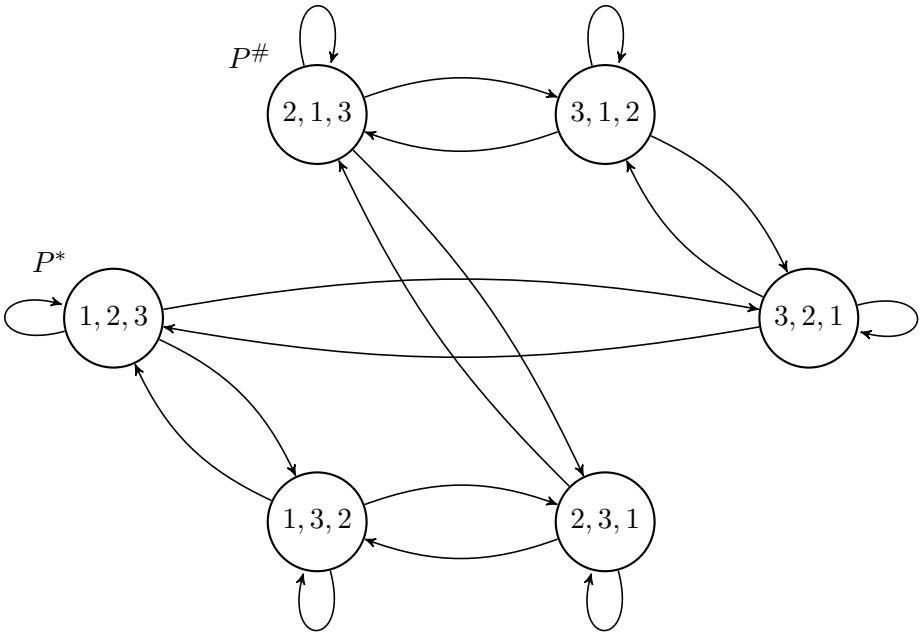


Figure 18: The Markov chain from Fig. 17 when one edge of the underlying graph is removed. State transitions that correspond to a swap of the first two elements in the assignment are not allowed.

choose the target distribution to be a Boltzmann distribution

$$\pi(a) = \frac{1}{Z} \exp\left[-\frac{1}{\tau}\Phi(a)\right] \tag{63}$$

with the normalization factor

$$Z = \sum_{\tilde{a}\in\mathcal{A}_P} \exp\left[-\frac{1}{\tau}\Phi(\tilde{a})\right] . \tag{64}$$

The variable τ is called the learning rate. Higher values of τ will lead to more stochastic exploration of the state space and less directed motion towards the optimal assignment P^*. As τ tends to zero, the probability for any state except the one with the minimum potential Φ go to zero. This is captured in the following definition of a stochastically stable state.

Definition A state $a \in \mathcal{A}$ of a Markov process is called stochastically stable if

$$\lim_{\tau\to 0} \pi_\tau(a) > 0 , \tag{65}$$

that is when the probability for being in that state is nonzero as the learning rate goes to zero (see [60]).

Note that the stochastically stable state is not necessarily unique. In fact, all assignments that have the same minimal cost as the assignment P^* are stochastically stable states, if they exist.

The Metropolis-Hastings algorithm defines the probability of a transition from state a to a' as the joint probability of the random proposition of this transition and the acceptance of it. In this case, the transition is a swap of two agents' targets. Therefore, proposing a transition is equivalent to choosing a pair of agents. Depending on the transition – i.e. the selected agents and their targets – an acceptance probability is computed. A random binary variable with this probability distribution then decides whether the agents accept the swap. The transition probability is

$$Pr[a \to a'] = q[a \to a'] \cdot \alpha[a \to a'] \,, \tag{66}$$

where q is the proposition probability and α is the acceptance probability. Because the edges of the graph are invariant to the state of the Markov chain and we randomly pick the swap pair e from a uniform distribution on the set of graph edges $\mathcal{E}$, the proposition distribution is uniform and we have

$$q[a \to a'] = q = const. \tag{67}$$

for all states $a \in \mathcal{A}_P$ and a' being any state that can be reached from a with one swap.

Given this proposition probability distribution, we have to design the acceptance probability α such that the stationary distribution of the Markov chain equals the target distribution π given in Eq. (63). We want to always accept a swap if it decreases the cost $\Phi(a)$ and randomly accept swaps that increase the cost. The probability for such a swap should decrease for higher cost increases. A common choice for the acceptance probability is the Metropolis choice

$$\alpha[a \to a'] = \min\left(1, \frac{\pi(a')}{\pi(a)} \frac{q[a' \to a]}{q[a \to a']}\right) = \min\left(1, \frac{\pi(a')}{\pi(a)}\right) \,. \tag{68}$$

This acceptance probability satisfies the detailed balance

$$\pi(a) Pr[a \to a'] = \pi(a') Pr[a' \to a] \,, \tag{69}$$

a condition for achieving the stationary distribution $\pi(.)$ with the Metropolis-Hastings algorithm. The derivation and the details of the algorithm are omitted here. They can be found in the original paper [38] and its generalization [24].

Lemma 3.2.14. *If the communication graph is connected, the assignment Markov chain is connected too. Further, every transition is reversible.*

Proof. Instead of a full proof, we will outline a way to swap any two elements. If the communication graph is connected, there is a path from any node to every other node. By swapping the leftmost element with the second, then the second element with the third an so forth, we can move the first element to the right of the path. Then by swapping the second rightmost element with the third element from the right and so forth back until the leftmost element, we have swapped the leftmost and rightmost elements with the elements in between being in their original position.

Since every swap is possible for a connected graph, any two assignments are connected on the Markov chain. □

Lemma 3.2.15. *The described Markov chain is ergodic, if* $Pr[a \to a'] > 0$ *for all transitions on the Markov chain and the communication graph is connected.*

Proof. A Markov chain is ergodic if every state can be reached from every other state with nonzero probability. If every transition in the Markov chain has nonzero probability and the Markov chain is connected (holds according to Lemma 3.2.14 because the communication graph is connected), there is a path with nonzero probability from every state in the Markov chain to every other state. □

Theorem 3.2.16. *The state* a^* *(the assignment* P^**) is a stochastically stable state of the Markov chain.*

Proof. The Markov chain is ergodic and the acceptance probability satisfies the detailed balance. Therefore according to [38], the stationary distribution of the Markov chain exists and is unique. It is equal to π. From Eq. (63) we can see that $\lim_{\tau \to 0} \pi(a) > 0$ if a is a minimizer

of Φ. Since P^* is defined as the minimizer of Δ, a^* is indeed a minimizer of Φ. The condition for stochastic stability from Section 3.2.2.2 is satisfied. □

When a swap of two agents' targets is proposed, it is accepted with probability α. Since we want to have a decentralized version of the Metropolis-Hastings algorithm, these two agents need to be able to compute the acceptance probability with local information only. We will show that the acceptance probability can be computed by either member of the pair if it has access to its own as well as its partner's information.

For the target distribution π given in Eq. (63), we have

$$\frac{\pi(a')}{\pi(a)} = \frac{\frac{1}{Z}\exp\left[-\frac{1}{\tau}\Phi(a')\right]}{\frac{1}{Z}\exp\left[-\frac{1}{\tau}\Phi(a)\right]} = \exp\left[-\frac{1}{\tau}(\Phi(a') - \Phi(a))\right] . \tag{70}$$

Let the transition from a to a' be a target swap on the edge $e = \{i, j\}$. Then a_e is the action that does not change the targets and a'_e is the bilateral action that swaps the targets of agents i and j. We have

$$\frac{\pi(a')}{\pi(a)} = \exp\left[-\frac{1}{\tau}(\Phi(a'_e, a_{-e}) - \Phi(a_e, a_{-e}))\right] . \tag{71}$$

Because of the alignment of the local and global potentials described in Eq. (61), this equals

$$\frac{\pi(a')}{\pi(a)} = \exp\left[-\frac{1}{\tau}(U_e(a'_e, a_{-e}) - U_e(a_e, a_{-e}))\right] . \tag{72}$$

As noted earlier, this requires only local information of the pair of agents. Therefore, the acceptance probability for a swap of two agents can be computed by either of those two agents if they share their information.

3.3 The Cost Function

To find the best assignment of pursuit agents to targets, we solved a minimization problem with a cost function. In this chapter we have been using the velocity mismatch of an agent and its target as a cost function for the target assignment. However, the value of a pursuit game is commonly represented by the time to capture or the distance between agent and target. We will discuss these cost functions and their relation to the velocity mismatch cost in Eq. (53).

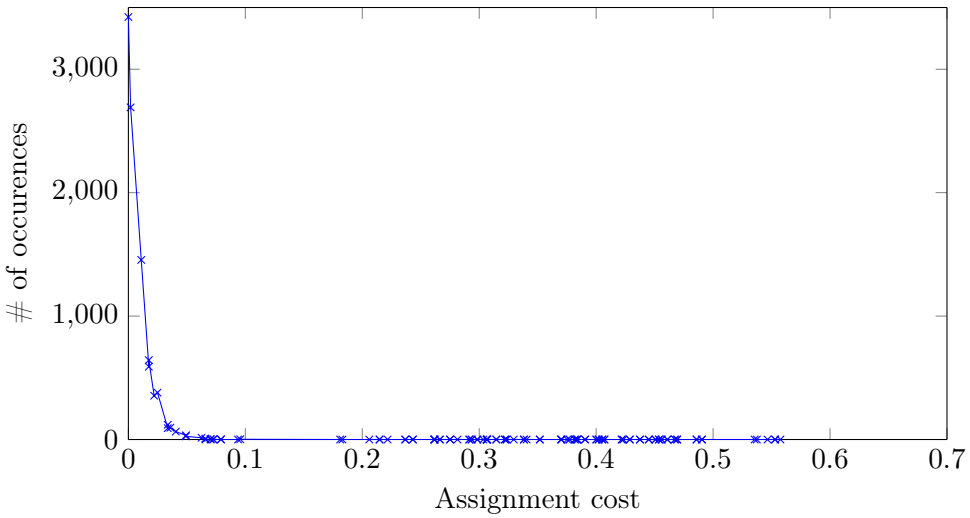


Figure 19: The assignment distribution in simulation for $N = 5$ and 10000 iterations. Each point corresponds to one assignment, they are ordered by their cost Φ. The learning rate is $\tau = 0.01$. It can be seen that most of the time is spent in assignments with a low cost.

The time to capture is the time that the agent needs until its position equals the target position, assuming perfect behavior of the agent as well as the target. Perfect behavior of the agent is the trajectory of actions over time that minimizes time to capture, whereas the target behaves such as to evade the agent as long as possible. Time to capture is not a useful measure when the target is faster than the agent – in that case, the time to capture is infinite. Moreover, we are interested in the results for large time horizons, as opposed to the transient-focused approach of time to capture.

Another common measure is the distance of an agent to its target, i.e.

$$d = \|x - y\| \ . \tag{73}$$

This can be evaluated at any given time. In a surveillance setting, this cost usually corresponds to the lack of observation quality, a distance of zero meaning perfect observation of the target. The value of the pursuit game is the limit of the distance as t goes to infinity, if that limit exists:

$$D = \lim_{t \to \infty} d(t) \ . \tag{74}$$

The distance of an agent to its target goes to zero over time if the agent can move faster than the target. If the agent is slower than the target, existence of the limit depends on the

arena.

For an unbounded space, an evader that is faster can infinitely increase the distance to the agent and the limit does not exist. In a closed, circular arena, the agent can achieve and maintain a certain distance from the target. In this case, the limit exists. An analytical expression has been found by [35] and is given in Section 3.3.2. In the more general case of arbitrary dynamics of agent and target and bounds on the state, the existence of the value has been shown [10]. We will limit our analysis to the simple integrator and a circular arena.

A special case is if the agent and the target have the same velocity. In this case, all cost functions are equivalent to some degree. This case will be mentioned in the following sections.

It should be noted that the limit of the distance is independent of the initial agent and target positions. It is a function of the ratio of the velocities of agent and target and the radius of the arena. We will look at the different cases of bounded and unbounded arenas in more detail now.

3.3.1 The Unbounded Pursuit

The pursuit in the unbounded plane has two important cases: The pursuer being faster than the target and therefore capture in finite time and the opposite case that leads to an infinite increase in the agent-target distance. The special case $\hat{u} = \hat{v}$ results in the distance being constant for all times.

If the agent is faster than the target, the limit of the distance for large t is zero. Therefore, the cost of this case is zero:

$$D(\hat{u} > \hat{v}) = 0 \, . \tag{75}$$

As stated before, the limit does not exist if the target is faster than the agent. In this case, the target can get arbitrarily far away from the agent. One could argue however, that an agent that is much slower than its target performs even worse than an agent that is almost as fast as the target. An agent with a velocity closer to that of the target would still not be able to keep up, but the increase in the distance would be slower. This can be expressed as the time derivative of the distance $\dot{d}$. Because the agent and the target are

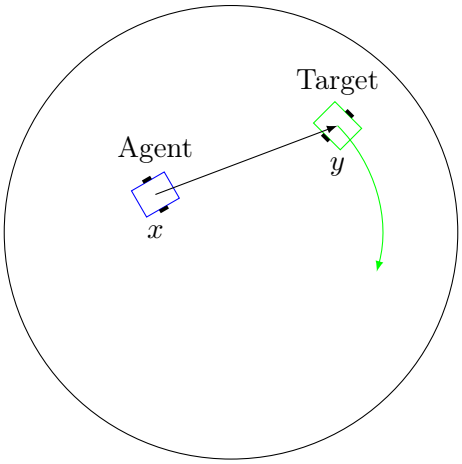


Figure 20: The pursuit in a circular arena.

moving in the same direction, the derivative of the vector norm can be expressed simply as

$$\dot{d} = \hat{v} - \hat{u} \; . \tag{76}$$

We can see the similarity to the velocity mismatch cost δ that we used earlier in the chapter. The distance still goes to infinity. Other costs might therefore make more sense, for instance counting the number of escaping targets. In this case, fast agents would be assigned to targets that are slower than them, while targets that are too fast would be abandoned or assigned to a slow agent.

3.3.2 The Circular Arena

The pursuit in a circular arena is known as the "Lion and Man" problem introduced by Rado (see [36]) and has been extensively studied by Flynn [18, 19] and Lewin [35] among others. Both were investigating problems equivalent to the one we study here: a pursuit of a target (a man) by an agent (the lion), where the agent is slower than or equally fast as the target and both are constrained to stay within the unit circle. This is sketched in Fig. 20. In the case of an real lion and a man this scenario is of course highly unlikely unless the lion is old and the man extremely fit. However, this assumption can be made because we are not biologists. Flynn proved the existence of a value of this game and found upper and lower bounds on the minimum distance the agent can achieve (and conversely the maximum distance that the target can maintain). He further provided a numerical solution for the

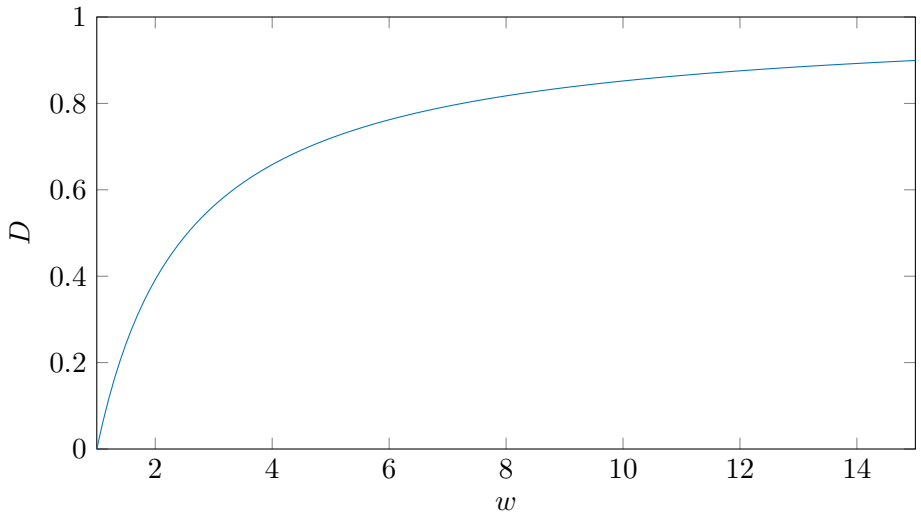


Figure 21: The value D of the circular pursuit game as a function of the velocity ratio w.

actual value of the game.

The analytical solution to this game is due to Lewin. The least upper bound on the distance that the evading target can maintain in the unit circle is a function of the ratio of the two players' speeds. With this ratio defined as

$$w = \frac{\hat{v}}{\hat{u}} \tag{77}$$

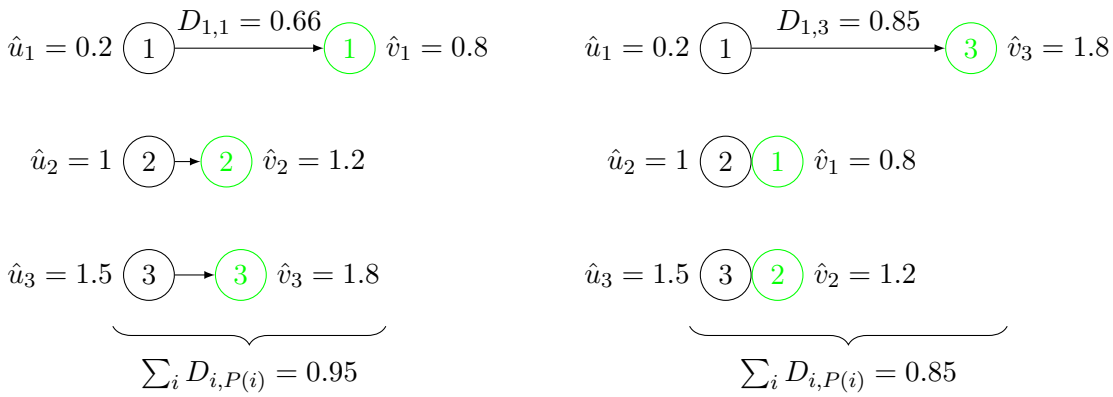
we have the limit of the distance [35]

$$D(w) = \frac{w^2 - 1}{w\sqrt{w^2 - 1} + w(\frac{\pi}{2} + \sin^{-1} w^{-1})} \,. \tag{78}$$

A plot of the distance limit as a function of the ratio w is displayed in Fig. 21.

We see that this value does not resemble the velocity mismatch cost. In fact, the distance cannot be more than 1 (the pursuer could simply wait in the center of the arena). For high ratios w, this limit is approached. This implies that, the higher w, the lower the effect of increasing the velocity ratio even further. In certain cases, this leads to interesting assignments. In Fig. 22, we show two different assignments for the same circular pursuit problem. The sum of the distance limits D is not minimal at the minimal velocity mismatch. Giving up on the fastest target by assigning it to the slowest agent and reassigning the faster agents to the slower targets leads to capture of two out of three targets.

This shows some resemblance to the proposition of abandoning targets that are too fast



(a) Minimal velocity mismatch. Every agent is slower than the assigned target.

(b) The slowest agent has an even greater distance to its target, but the other agents can achieve capture.

Figure 22: Two different assignments in the circular arena pursuit. The overall tracking in the right assignment is better, even though the velocity mismatch is higher.

in the unbounded pursuit. Albeit, the reasons are different. In the unbounded pursuit we can give up on a target if it is too fast even for the fastest available agent. Assigning it to a slow agent does not change the fact that it can escape. In the circular arena however, this result comes from the fact that the maximal distance that a target can achieve is bounded by the radius of the arena. This means that the cost of assigning a very slow agent to a fast target is bounded and often outweighed by the cumulative cost saving of assigning the faster agents to slower targets.

It should be noted however that the optimal assignment for this cost equals the optimal assignment P^* for Δ if the teams have the same velocities. In fact, in this case all cost functions are equivalent in the sense that the optimal assignment is the same.

3.4 Optimal Team Assembly

Similarly to Chapter 2, we are again interested in the optimal team assembly given a task. In the case of the multi-agent multi-target pursuit, the task is defined by a group of target agents or their velocities. The agent team can also be represented as a vector of movement velocities. Therefore we are looking for a mapping

$$Q^* : \mathbb{R}_+^N \mapsto \mathbb{R}_+^N \tag{79}$$

of the targets' velocity vector $\hat{v}$ to the agents' velocity vector $\hat{u}^*$ such that $\hat{u}^*$ minimizes a cost

$$J(\hat{u}^*, \hat{v}) = \min_{u \in \mathbb{R}_+^N} J(u, \hat{v}) \ . \tag{80}$$

The cost J represents our objectives. On the one hand, the pursuit quality as measured by Δ_0 should be as good as possible. We introduce the pursuit cost $\Delta_0(P)$ to be the cost Δ without the cost for an agent being too fast for its target, i.e.

$$\Delta_0(P, \hat{u}) = \sum_{i \in \mathcal{N}} \delta_0^2(\hat{u}_i, \hat{v}_{P(i)}) \tag{81}$$

with the pairwise cost

$$\delta_0^2(\hat{u}_i, \hat{v}_{P(i)}) = \begin{cases} (\hat{u}_i - \hat{v}_{P(i)})^2, & \hat{u}_i < \hat{v}_{P(i)} \\ 0, & \hat{u}_i > \hat{v}_{P(i)} \end{cases} \ . \tag{82}$$

This is evaluated for the best possible assignment and we define the pursuit cost to be $\Delta_0^*(\hat{u}) = \Delta_0(P^*, \hat{u})$. The dependence of Δ on $\hat{u}$ is not new, but before $\hat{u}$ was not a variable.

On the other hand, the agents should consume as little energy as possible. We introduce an energy-usage cost $T(\hat{u})$ for this purpose. We compose J in the following way:

$$J(\hat{u}, \hat{v}) = \Delta_0^*(\hat{u}, \hat{v}) + R \cdot T(\hat{u}) \ , \tag{83}$$

where the coefficient $R \in \mathbb{R}_+$ is a weighting term for the energy-cost.

T is a cost of a potential. Unlike the energy-term in the cost c in the previous chapter, it does not directly correspond to motion-induced energy consumption. Rather, it is a cost of building or using a robot with the capability of moving fast. It could correspond to actual monetary cost of building such a robot. Unsurprisingly, a robot that can only move slow can be built more cheaply than a robot that satisfies high requirements on the maximal velocity. Further, robots that have the capability of moving very fast have to be designed for higher accelerations and therefore higher forces, typically requiring more durable and massive electrical and mechanical parts. This leads to a higher total weight of the robot, which in turn increases energy consumption even when the robot is not exhausting its velocity potential. This means that T has to be an increasing function of the velocities $\hat{u}_i$.

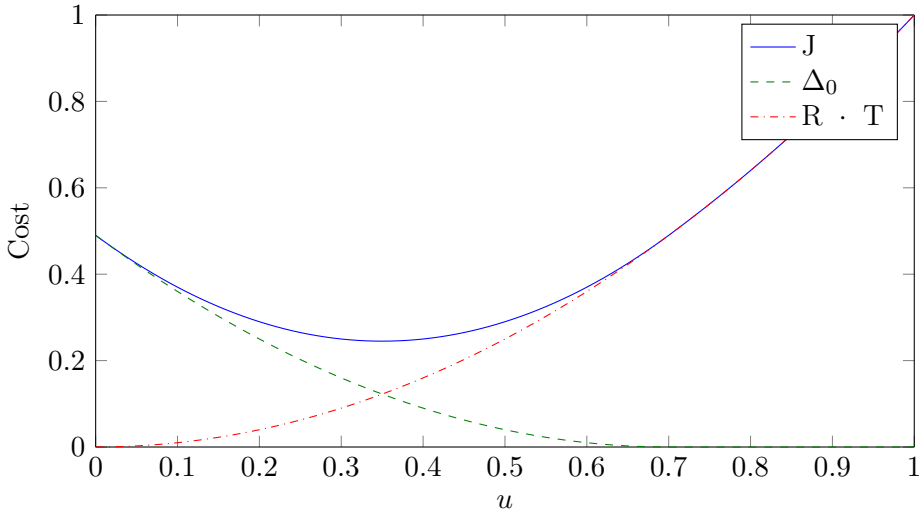


Figure 23: The cost function J for a single agent. The energy-cost is chosen as $T(\hat{u}) = \hat{u}^2$ and $R = 0.3$. The target velocity is $\hat{v} = 0.7$. The optimal agent velocity is $\hat{u}^* = 0.54$.

We will assume that it is the sum of each single agent's cost

$$T(\hat{u}) = \sum_{i \in \mathcal{N}} T_i(\hat{u}_i) \tag{84}$$

where T_i is a increasing function of $\hat{u}_i$.

Naturally, a group of agents with velocities that match the targets' minimizes the pursuit cost Δ_0^*. To reduce the cost T however, the velocities should be as low as possible. The solution to the minimization problem in Eq. (80) is a compromise between the two competing goals. It can be interpreted as sacrificing some of the pursuit quality (or surveillance quality in a target observation setting) for a reduction in robot cost.

A plot of the costs Δ_0, T and J for exemplary values is displayed in Fig. 23. It can be seen that the velocity that minimizes Δ_0 is $\hat{u} = \hat{v} = 0.7$ or anything higher than that. However, after adding the weighted energy-term T, the optimal velocity is below that at $\hat{u}^* = 0.54$. We will show in simulation that the tracking results are still acceptable in this case.

An example of a pursuit of a fast target by a slower agent can be seen in Fig. 24. The plot shows the distance between the agent and its target over time. The target is not actively evading the agent, but rather roaming around the bounded arena, driving curves with random curvature at constant maximal velocity. Even though the agent is slower than

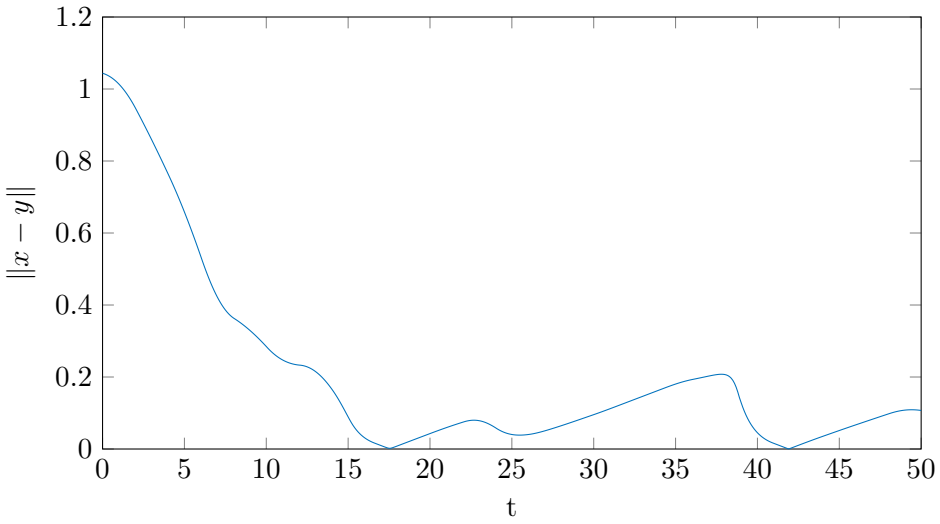


Figure 24: The distance between an agent and its target over time in a bounded arena. The agent is slower than the target with $\hat{u} = 0.8 \cdot \hat{v}$.

the target with $\hat{u} = 0.8 \cdot \hat{v}$, the distance stays close to zero after an initial decrease. This illustrates that the sacrifice in surveillance quality is usually not as big as expected from the pursuit evasion game.

If all agents can have different velocities, their optimal velocities can be computed independently from each other and we have

$$\hat{u}_i^* = \underset{\hat{u}_i \in \mathbb{R}_+}{\operatorname{argmin}} \left\{ \delta_0^2(\hat{u}_i, \hat{v}_{P(i)}) + R \cdot T_i(\hat{u}_i) \right\} \ . \tag{85}$$

Even though the optimal assignment P^* depends on the velocities of all agents, the optimal values for the velocities conserve the assignment. In other words: If we calculate the optimal velocities for any assignment P, then the resulting velocity vector will make this assignment the optimal assignment. Note that if we choose another assignment than P^*, the order of the agents' velocities also changes. If we reorder them by their maximum speed, the optimal assignment is P^* again.

3.4.1 Example: A Team of RC-Racers

As an example for the assembly cost T of the robotic team, we will build a pursuit team that minimizes the velocity mismatch cost Δ_0 with a given budget for the electrical motors of the robots. The general platform of the robots is given and we have to buy a set of electrical

motors to fit on the robots and drive them. Note that we do not actually build robots, but use this as an example for how an optimal team might be assembled.

In this approach, we can find a few differences to the optimal team composition described before. For one, we don't have a continuous (i.e. infinite) set of different maximal velocities we can choose our motors from. Instead, we need to pick from the set of motors that are offered on the market, preferably from just one manufacturer to simplify the robots' assembly and maintenance. So we have a set of motors, each of them with a price tag and a maximal output power. We call this set $\mathcal{U}_{\text{mot}}$. The other difference is that we use the cost T only as a constraint instead of adding it to the cost function. Where before the task was to find a team that is a compromise between pursuit quality and motor retail cost, we now search for the team with the highest pursuit quality while not exceeding a certain cost $T_{\max}$.

In this example, we use the RC-motors offered by the manufacturer Horizon Hobby. Specifically we choose brushless out- and inrunner motors from the "Park Series" and the "Power Series" ranging from 30 W to 2700 W at prices between \$30 and \$150. We gathered the retail prices and the power ratings for all offered motors. To determine the maximal speeds of the robots, we can compute the air resistance with the drag equation

$$F_D = \frac{1}{2}\rho u^2 C_D A \,, \tag{86}$$

where F_D is the drag (the resistance force), ρ is the density of the fluid (in this case air), u is the velocity of the object and A is its reference area, that is the area of the projection of the robot on a plane orthogonal to the movement direction. Given the drag, we can determine the necessary power to overcome the air resistance with

$$P_D = F_D \cdot u = \frac{1}{2}\rho u^3 C_D A \,. \tag{87}$$

Solving this equation for the velocity u gives us the maximum velocity of a robot for a specific motor's power output when we only take air resistance into account:

$$\hat{u} = \left(\frac{2P_D}{\rho C_D A}\right)^{\frac{1}{3}} \,. \tag{88}$$

At normal temperature and pressure (20 °C and 101.325 kPa), dry air has a density of $\rho = 1.2041\,\text{Kg/m}^3$. Further, we assume a drag coefficient of $C_D = 0.26$ (corresponds to the

Table 1: Prices of RC-motors with the corresponding power values and top speeds.

Price [$]	Power [W]	Top speed [m/s]	Price [$]	Power [W]	Top speed [m/s]
30.00	30	8.67	70.00	550	22.86
32.00	55	10.61	75.00	700	24.77
38.00	85	12.27	90.00	800	25.90
40.00	150	14.83	110.00	1200	29.65
45.00	225	16.97	130.00	1800	33.94
46.00	325	19.18	140.00	2000	35.15
60.00	375	20.12	150.00	2700	38.85
65.00	425	20.98			

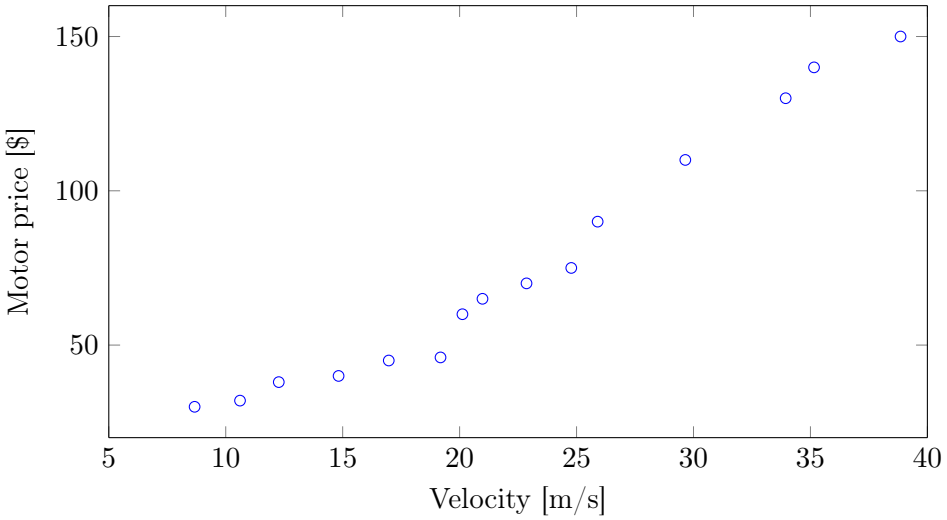


Figure 25: The cost of the RC motors plotted over the maximum velocity.

drag coefficient of the BMW i8, a sports car) and a reference area of $A = 0.1\,\mathrm{m}^2$. These values are chosen arbitrarily. Since the power values given on the manufacturer's website are electrical power input values and no efficiency values are available, we assume an efficiency of 85% for every motor. With these values, we can compute the maximum velocity for every motor. The retail prices, the power inputs and the resulting top speeds of all motors are given in Table 1. A plot of the retail prices over the maximum speed, i.e. the cost $T(\hat{u})$, is given in Fig. 25. We can see a roughly quadratic relation of the two variables.

With this cost function for discrete points of $\hat{u}$, we want to solve a discrete optimization problem with constraints. The task is the following: for given $\hat{v}$, minimize the cost Δ_0 while not exceeding the budget, i.e. $T(\hat{u}) \leq T_{\max}$. The values $\hat{u}_i$ may only be chosen from a discrete set $\mathcal{U}_{\mathrm{mot}}$. This problem can be stated as an integer programming problem, where

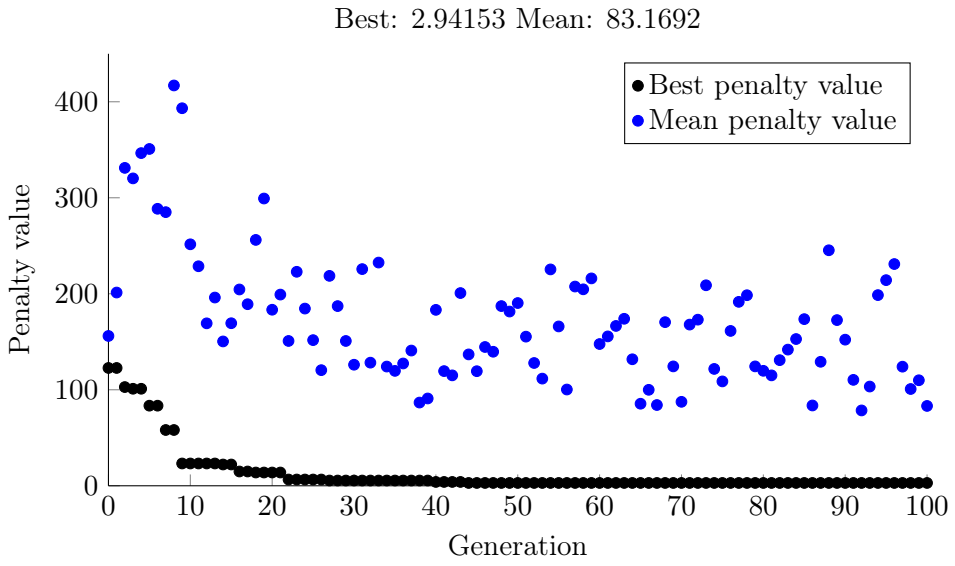


Figure 26: Plot of the mean and best cost Δ_0 of the population of the genetic algorithm for 100 iterations.

the optimization variable is a vector of integers that point to values in $\mathcal{U}_{\text{mot}}$. Because the total number of possible compositions is $|\mathcal{U}|^N$ (with $|\mathcal{U}|$ being the cardinality of $\mathcal{U}$), we use an optimization algorithm to search for the optimal team.

We choose to use a genetic algorithm for this problem, because it is suitable for integer programs and the notion of crossover and mutation is clearly appropriate for the used optimization variable. The results of this approach are generally very good. Also, the computation time is rather short, with the algorithm finding a near-optimal solution within a few seconds on current computer hardware.

Example 3.4.1. *Let the vector of evader velocities be* $\hat{v} = (10, 13.5, 18, 20, 21, 26)$. *We try to find the best motors for the pursuer team given a budget of \$300. The genetic algorithm is initialized with the cost function* Δ_0, *a number of six variables and the constraint* $T(\hat{u}) \leq T_{\max} = 300$. *Within a few iterations, a very good solution is found. The evolution of the solutions is displayed in Fig. 26. There we can see that the mean cost of the population decreases for more iterations. The population size is 60, therefore 60 evaluations of the cost function per iteration are necessary.*

The solution to the problem is $\hat{u} = (10.61, 14.83, 19.18, 19.18, 20.12, 24.77)$ *at a cost of* $T = 299$. *The velocity mismatch cost is* $\Delta_0(\hat{u}, \hat{v}) = 2.94$. *It is noteworthy that if we use the*

Table 2: The results of the genetic algorithm for three different cost functions.

Targets	$\hat{v}_1$	$\hat{v}_2$	$\hat{v}_3$	$\hat{v}_4$	$\hat{v}_5$	$\hat{v}_6$		
	10	13.5	18	20	21	26		
Cost	$\hat{u}_1$	$\hat{u}_2$	$\hat{u}_3$	$\hat{u}_4$	$\hat{u}_5$	$\hat{u}_6$	Retail Price	Pursuit Cost
Δ_0	10.61	14.83	19.18	19.18	20.12	24.77	\$299	$\Delta_0 = 2.94$
D	10.61	14.83	19.18	19.18	20.12	24.77	\$299	$D = 0.084$
N_{esc}	14.83	16.97	19.18	20.98	22.86	*10.61*	\$298	$N_{esc} = 1$

cost D – the limit of the distance in the circular pursuit as described in Section 3.3.2 – the result is the same as for the cost Δ_0.

However, using the number of escaping targets in the unbounded pursuit – as suggested in Section 3.3.1 – yields a different result. It can be seen that the solution to the original problem produces three agents that can capture their targets and three agents that are slower. In fact, the number of escaping targets can be reduced to one if this one target is abandoned. One solution is $\hat{u}' = (14.83, 16.97, 19.18, 20.98, 22.86, 10.61)$. In this case, all agents are faster than their target, with the exception of the last one. These results are summarized in Table 2.

Chapter IV

ROBOTIC IMPLEMENTATION

The theoretical results from Chapters 2 and 3 have been validated in simulation. To further demonstrate the practical feasibility of the developed methods, they have also been implemented on a physical robotic platform.

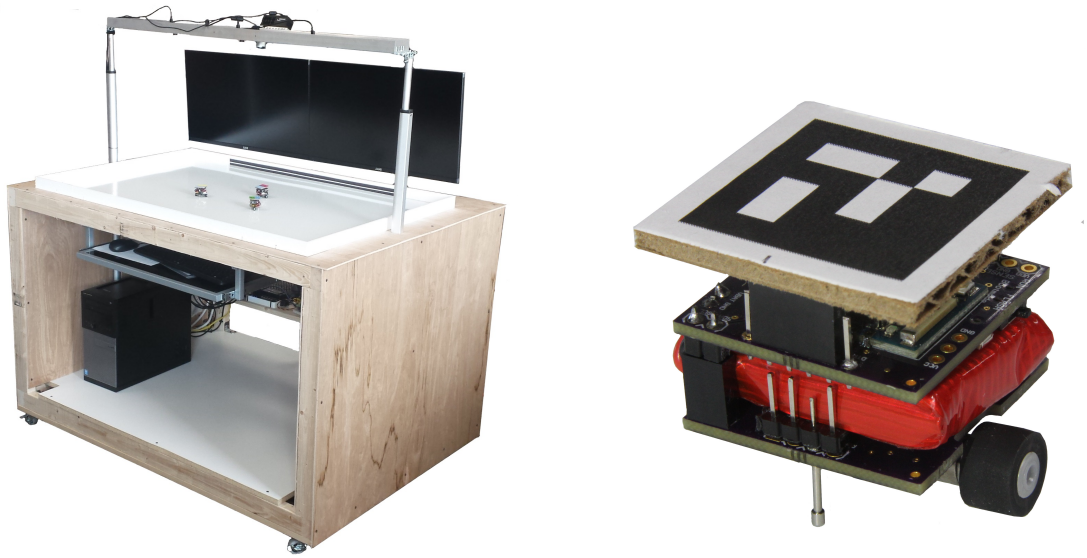
4.1 The Robotarium

The Robotarium is a remote-access multi-robot testbed designed to test and run networked control algorithms on real robots. It was developed by the GRITS-lab at Georgia Tech as a way for researchers of all disciplines to run their models on a real robotic platform. The prototype of the Robotarium uses a table as the arena for the robots, the so-called GRITSBots. Pictures of the Robotarium and the GRITSBots are displayed in Fig. 27. The Robotarium and its use are described in more detail in [45].

The GRITSBots are low-cost differential drive robots with an IEEE 802.11 antenna, a computing unit and a battery pack. The robots have identifying tags printed on top and their position is acquired by a visual tracking system above the table. A server tracks the positions of the robots and communicates with them using an IEEE 802.11 connection. The control laws are implemented on the server, which computes the control inputs of the robots and remote controls their physical actions. This setup allows for centralized as well as distributed control laws, with the latter still being executed on the server. An overhead projector that projects onto the table surface allows for visual additions to the Robotarium arena and can provide additional information as well as simulated changes to the environment, e.g. obstacles.

4.2 Nonlinear Robot Dynamics

In the preceding chapters, all methods have been developed for linear, single-integrator robot models. This assumption is not valid for the two-wheeled GRITSBots that are used



(a) The Robotarium prototype. (b) One of the GRITSBots.

Figure 27: The Robotarium and one of its GRITSBots [1].

in the experimental setup. Instead, the robots can be modeled with the so-called unicycle dynamics

$$\begin{aligned} \dot{x}_1 &= v\cos\theta \\ \dot{x}_2 &= v\sin\theta \\ \dot{\theta} &= \omega \end{aligned} \tag{89}$$

with the inputs v and ω denoting velocity and angular velocity respectively.

We can then transform the inputs v and ω to the angular velocities of the two wheels of the robot. If the distance of the wheels from the center of the robot is r and the radius of the wheels is r_w, the transformation for the wheel speeds ω_L of the left and ω_R of the right wheel is

$$\begin{aligned} \omega_L &= \frac{1}{r_w}(v - r\omega) \ , \\ \omega_R &= \frac{1}{r_w}(v + r\omega) \ . \end{aligned} \tag{90}$$

The unicycle model of a two-wheeled robot is displayed in Fig. 28.

The mechanics of the unicycle model are non-holonomic, because the constraints on the motion cannot be derived from position constraints. The non-holonomic constraint here is

[1] The images "Robotarium" and "GRITSBot" are the property of Daniel Pickem. The owner granted permission to use them in this work.

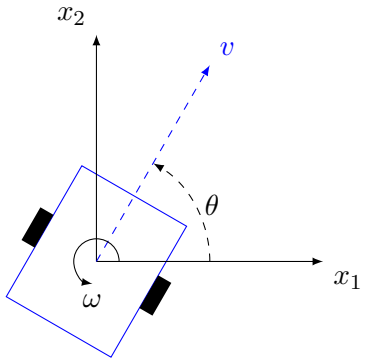


Figure 28: A simple unicycle model.

the restriction of motion to the direction the wheels are facing. No direct sideways motion is possible. This also leads to the uncontrollability of the linearized system around a state $x^* = [x_1^*, x_2^*, \theta^*]^{\mathrm{T}}$:

$$\Delta\dot{x} = \begin{pmatrix} \cos\theta^* & 0 \\ \sin\theta^* & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix} . \tag{91}$$

The first two states' linearized equations are never independently controllable (note that θ^* is a constant).

For these reasons we have to use nonlinear control laws that enable us to overcome the non-holonomic constraints. Path planning and control of two-wheeled robots have been extensively studied in the past [28, 34, 33, 54, 53].

4.3 Target Tracking Controller

A target tracking controller for the differential-drive robots should have the following properties:

- Globally stable error dynamics.
- Maintain a safety distance from target robot to avoid collisions.
- Still function with input saturation.
- Reach target quickly.

While the last goal is optional, in a surveillance setting it is desirable to keep the time that the target is unobserved as short as possible. Further, we would like our agents not only to keep some distance to their targets, but also to every other robot or obstacle. We will use barrier certificates to achieve this goal.

4.3.1 Collision Avoidance

To achieve collision avoidance not only between an agent and its target, we apply safety barrier certificates introduced in [6]. Without having to change the original robot controllers, the collision avoidance is wrapped around the user-defined control law on a low level. It modifies the control input to the system in a minimally invasive way if a collision of two agents would otherwise be unavoidable.

This is done by defining a safety radius around each agent that no other agent may enter. If a collision would happen with the original input u, a new control input u^* is computed that does not lead to a collision. The new control input u^* is the least invasive collision free input, i.e. the difference between u and u^* is minimal. To compute u^*, a quadratic program has to be solved in every time step.

4.3.2 Tracking in Polar Coordinates

We use the controller proposed in [33] with an extension for better performance when the target is moving. It drives the agent's distance to the target robot to a specified value and its heading towards the target. The states of the system are transformed to a relative position and heading, where r is the distance of the agent from the target position and ϕ is the difference between the facing angle of the agent and the angle of the line connecting the agent and the target. We have the position error

$$e = P - x \tag{92}$$

and with this the states

$$\begin{aligned} r &= \|e\| \,, \\ \phi &= \theta - \tan^{-1} \frac{e_2}{e_1} \,. \end{aligned} \tag{93}$$

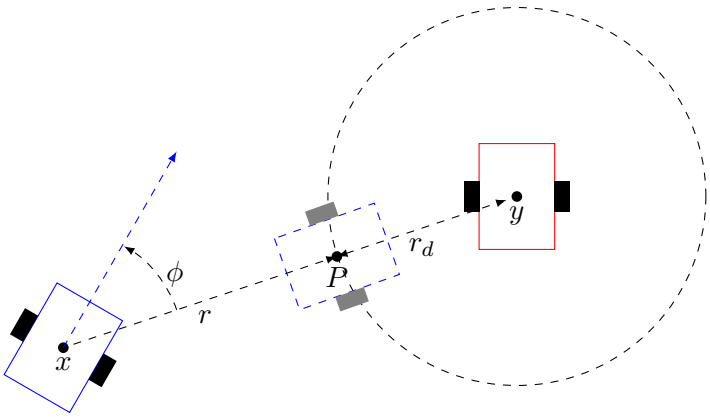


Figure 29: A visualization of the control goal of the polar coordinate controller. The agent on the left tries to reach the distance r_d to the target agent on the right. The desired position of the left agent is drawn dashed.

A visualization of the goal of this controller is displayed in Fig. 29. The point P is the target position for the agent and r is the distance between x and P. The target P is at the desired distance r_d from y.

According to [33], the robot dynamics are

$$\begin{aligned} \dot{r} &= -v\cos\phi\ , \\ \dot{\phi} &= \omega + \frac{v}{r}\sin\phi\ . \end{aligned} \tag{94}$$

The system is stabilized by the feedback law

$$\begin{aligned} v_p &= k_1 r\cos\phi\ , \\ \omega_p &= -k_1\sin\phi\cos\phi - k_2\phi\ . \end{aligned} \tag{95}$$

This model does not take into account the movement of the target agent y. To improve its performance for moving targets we extend the model, assuming we know the velocity vector of the target at any time.

4.3.2.1 Moving target tracking improvement

Even though the original controller is designed in polar coordinates, we will use Cartesian coordinates for this extension. When no coordinate frame is specified, vectors are represented in the global coordinate frame W.

First, the target movement is transformed from the global coordinate frame W to the agent's local coordinate frame A. This is done by rotating it by θ, the orientation of the

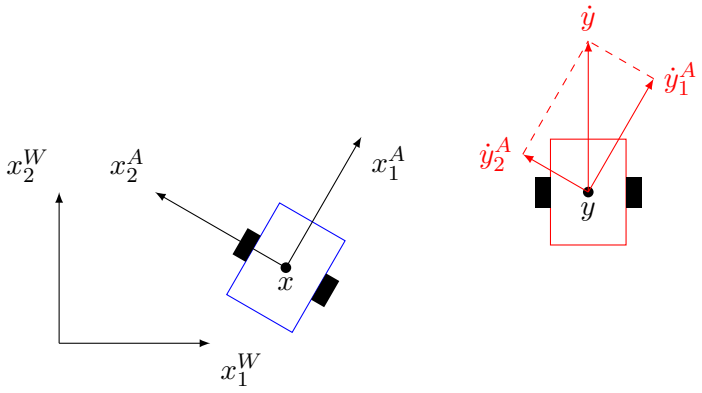


Figure 30: The global and local coordinate frames of the agent.

agent x in the global coordinate frame:

$$\dot{y}^A = \begin{pmatrix} \dot{y}_1^A \\ \dot{y}_2^A \end{pmatrix} = \boldsymbol{R}_\theta \dot{y}^W \ , \tag{96}$$

where the rotation matrix is

$$\boldsymbol{R}_\theta = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \ . \tag{97}$$

This is represented in Fig. 30.

To account for the target movement, we add terms to our control inputs that mitigate the effects of target movement. If the target moves along the frontal axis of the robot, this movement can easily be canceled out by adding the velocity in this direction to the velocity input as the term

$$v_d = \dot{y}_1^A \ . \tag{98}$$

For target movement along the orthogonal axis x_2^A, the movement constraints of the agent do not allow this. Instead, we adjust the heading direction of the agent to match the movement of the target. We transform the position error vector e into the local coordinate frame with $e^A = \boldsymbol{R}_\theta e^W$. A movement of the target along the orthogonal axis of the agent corresponds to an angular velocity of

$$\omega_d = e_1^A \cdot \dot{y}_2^A \tag{99}$$

of the agent.

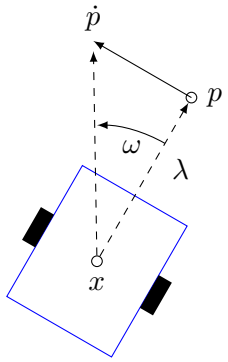


Figure 31: The transformation from single integrator to unicycle dynamics. In this example, the transformation leads to a purely rotational movement.

With the feedback terms from Eqs. (95), (98) and (99), the overall control law now is

$$\begin{aligned} v &= v_p + v_d = k_1 \cdot r \cdot \cos\phi + \dot{y}_1^A \ , \\ \omega &= \omega_p + \omega_d = -k_1 \cdot \sin\phi \cdot \cos\phi - k_2 \cdot \phi + e_1^A \cdot \dot{y}_2^A \ . \end{aligned} \tag{100}$$

4.3.3 Single Integrator Transformation

Because the control laws in Chapter 2 have been developed for single integrator dynamics, we prefer to continue using these in the implementation. However, the robots are described with the unicycle dynamics. Given a control law $u_{int}(x)$ for the single integrator dynamics

$$\dot{p} = u_{int} \tag{101}$$

where $p \in \mathbb{R}^2$, we need a transformation from u_{int} to a unicycle input u that leads to a corresponding behavior of the nonlinear robot. This transformation is achieved by placing a reference point at a distance λ in front of the robot and having the robot follow this point. A sketch of this can be seen in Fig. 31.

The position of the reference point p with respect to the robot is

$$p = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \lambda \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} \ . \tag{102}$$

Differentiating this term with respect to time yields

$$\dot{p} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} + \lambda \begin{pmatrix} -\omega\sin\theta \\ \omega\cos\theta \end{pmatrix} = \begin{pmatrix} u_s\cos\theta - \lambda\omega\sin\theta \\ u_s\sin\theta + \lambda\omega\cos\theta \end{pmatrix} \tag{103}$$

where u_s and ω are the inputs for velocity and rotational velocity of the unicycle. Inserting $\dot{p} = u_{int}$ and rewriting this equation leads us to the transformation

$$u_{int} = \begin{pmatrix} \cos\theta & -\lambda \sin\theta \\ \sin\theta & \lambda \cos\theta \end{pmatrix} \begin{pmatrix} u_s \\ \omega \end{pmatrix} . \tag{104}$$

Note that $u_{int} \in \mathbb{R}^2$. This transformation is a diffeomorphism. Through inversion, we get the desired transformation from u_{int} to the unicycle inputs:

$$\begin{pmatrix} u_s \\ \omega \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\frac{1}{\lambda}\sin\theta & \frac{1}{\lambda}\cos\theta \end{pmatrix} u_{int} . \tag{105}$$

This inverse exists for $\lambda > 0$. We use this transformation in the assignment problem to make the robots follow the single integrator trajectories.

4.4 Assignment

The assignment algorithm has been implemented in MATLAB in the Robotarium simulator. Because the GRITSBots are modeled as unicycles, we use the transformation described in Section 4.3.3 to transform the single integrator control input to the unicycle inputs v and ω. We further employ collision avoidance by using the technique described in Section 4.3.1 on the single integrator dynamics. The visual output of the Robotarium simulator for the assignment problem can be seen in Fig. 32.

Additional information is displayed via the overhead projection system. We use colored circles around the robots that show their velocity (blue circles denote fast robots, red circles are slow). The target positions are marked as green crosses. The assignment of agents to target is visualized with lines, using the same color coding as the robots. An example of the projected image is shown in Fig. 33.

Switching from the simulator to the real robots only requires minor adjustments to the MATLAB scripts. As expected, the experiment runs smoothly on the Robotarium. A photo of the agents performing the task can be seen in Fig. 34.

Collision avoidance leads to rather small changes in the behavior of the robots and increases the distance driven. The trajectories of the robots are visible in Fig. 35. This is not considered in the assignment algorithm and therefore increases the cost beyond the value

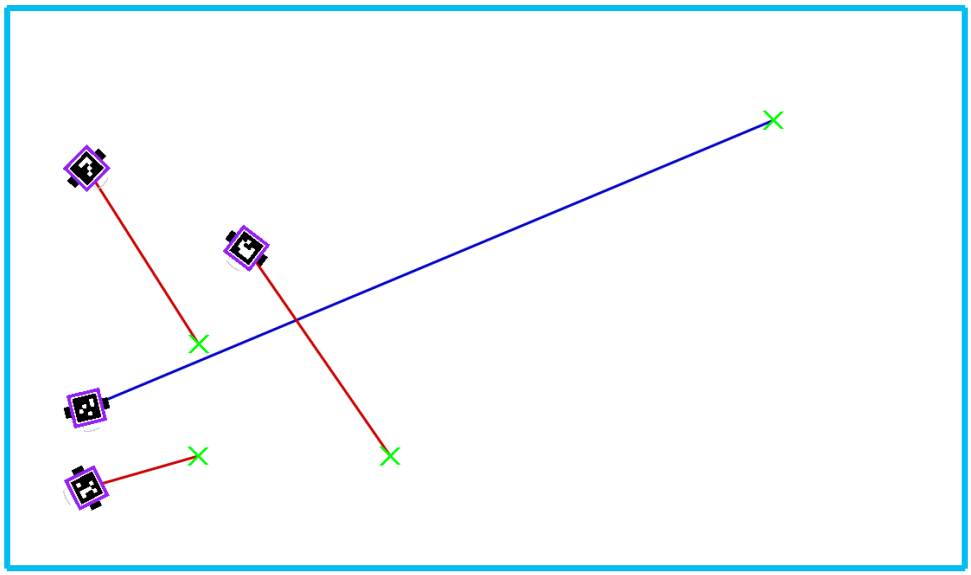


Figure 32: The assignment problem in the Robotarium simulator.

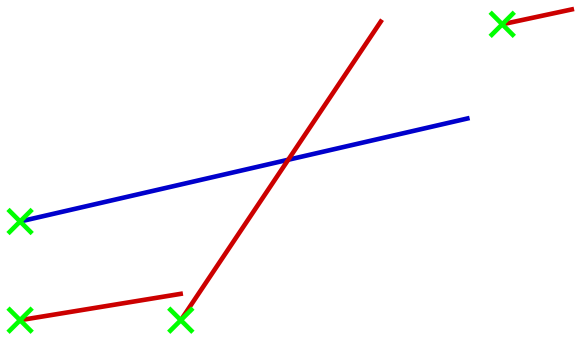


Figure 33: The overhead projection used for the assignment problem. The colored lines are displayed between the robots and their assigned targets, the crosses mark the target positions.

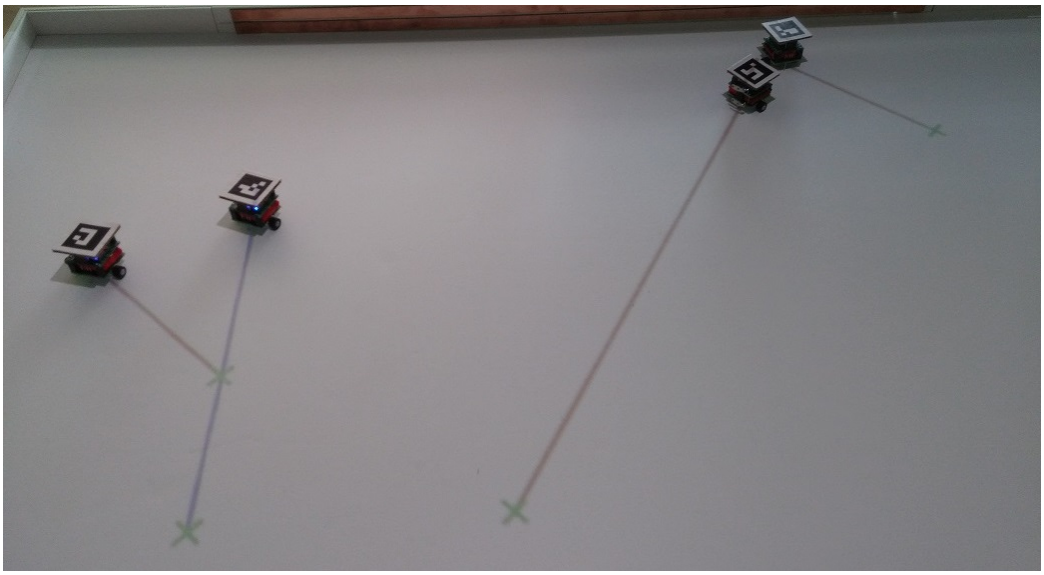


Figure 34: A photo of the GRITSBots performing the assignment algorithm.

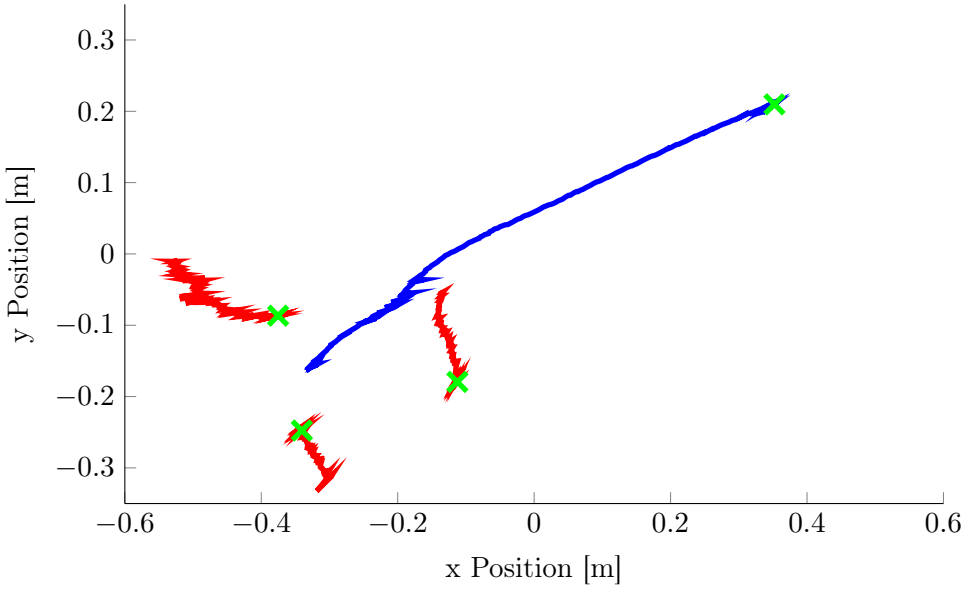


Figure 35: The trajectories of the robots in the assignment problem.

proposed by the algorithm. In crowded environments, it would be interesting to examine the effect of collision avoidance on the quality of the assignment. It is possible that seemingly suboptimal assignments are better for the real robots if they lead to less crossed paths.

4.5 Pursuit

Similarly to the assignment, the target pursuit was first implemented on the Robotarium simulator. We use the GRITSBots as pursuers as well as evaders. The control law described in Section 4.3.2 considers the unicycle dynamics, so there is no need for the diffeomorphism from Section 4.3.3. However, the collision avoidance algorithm is implemented for the single integrator dynamics. Although the tracking controller ensures that the pursuer does not collide with its target robot, collisions between any other robots cannot be ruled out, so we need to add collision avoidance. By transforming the unicycle dynamics to the single integrator dynamics, then applying the collision avoidance and transforming back to the unicycle inputs we solve the collision avoidance task. An example configuration of the pursuit in the simulator can be seen in Fig. 36.

The overhead projection is similar to that in the assignment problem. However, instead of just 2 different colors for the agents, we display a gradient from red (slowest agent) to blue (fastest agent). The target robots are marked with green circles, where the brightness of the color corresponds to that target's speed. The dark green circle corresponds to the slowest target, the light green circle to the fastest.

Similarly to the assignment problem, the implementation of the pursuit problem on real robots required minor adjustments. A photo of the robots performing the target pursuit can be seen in Fig. 38. Because the arena is bounded, a function was introduced that prevents collisions of the robots with the wall of the Robotarium.

A large number of robots in the arena complicates the task because collision avoidance leads to many changes in the robots' trajectories. This is even more apparent since the robots are only coordinated in groups of two (pursuer and evader) and there is no general framework that coordinates all robots together like in formation control. This means that the paths of robots will cross many times, leading to collision avoidance maneuvers.

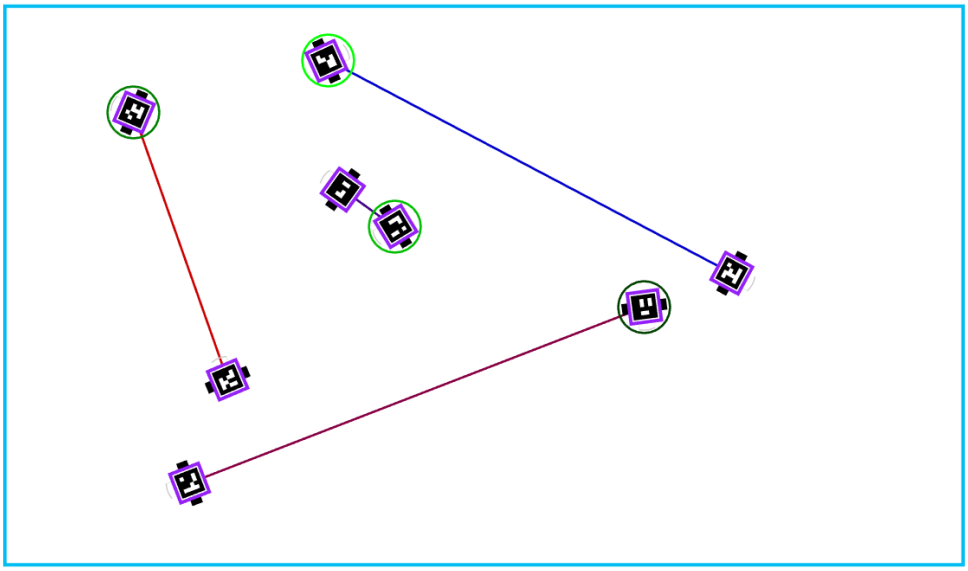


Figure 36: The pursuit problem in the Robotarium simulator. The green circles denote the target agents.

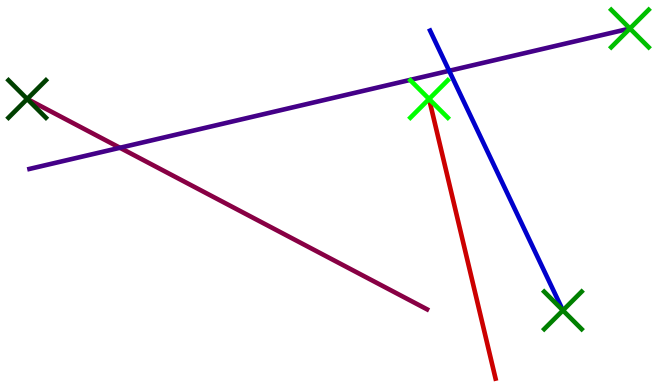


Figure 37: The overhead projection used for the pursuit problem. The red to blue colored lines are displayed between pursuer and evader, the green crosses are projected on the target robots.

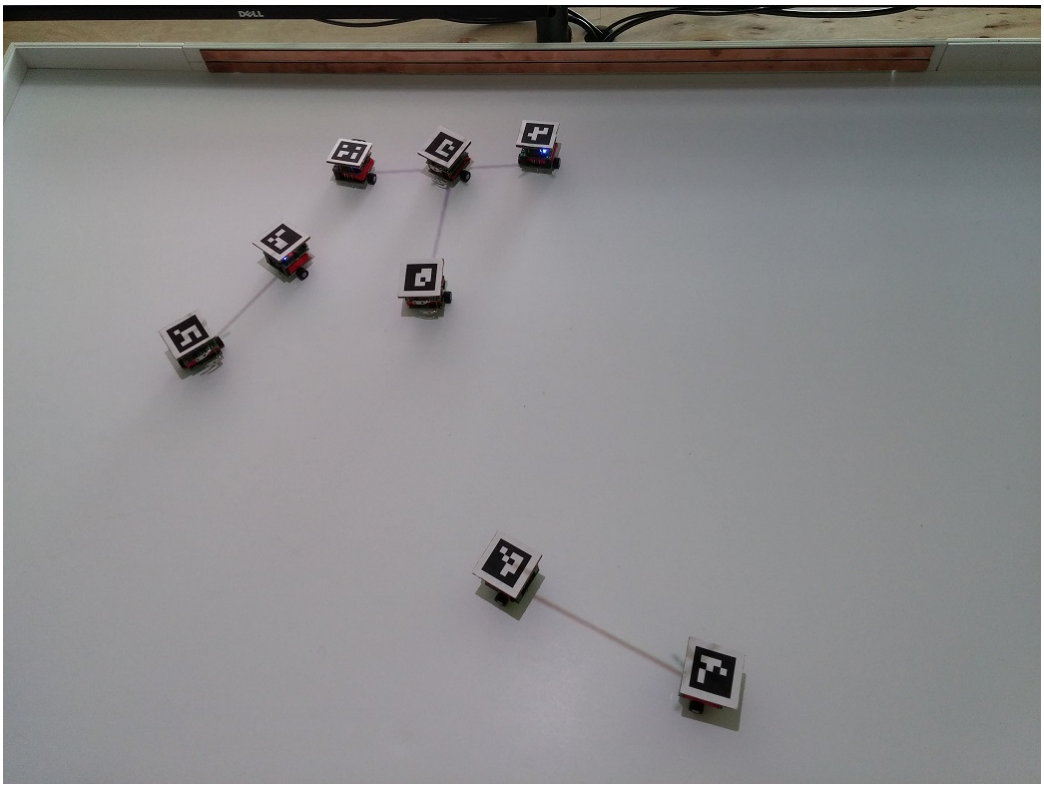


Figure 38: A photo of the robots performing the pursuit algorithm. The lines are projected onto the table by an overhead projector.

Chapter V

CONCLUSION

In this work, we investigated the application of temporally heterogeneous teams to two distinct problems. We showed ways to incorporate slowness into robotic teams and the benefits and drawbacks of slow team members.

One of the investigated topics was a target assignment. By choosing a cost measure that depends both on the distance to a target and the energy consumption needed to reach that target, we introduced an optimization problem. We then solved the task of assembling the optimal team of slow and fast agents to solve the assignment problem with minimal cost. To complement this result we derived an algorithmic method for the optimization of the agents' movement velocities, thereby creating a tool to design agents, assemble a team and solve the assignment problem for given initial and target positions. We showed the practicality of the developed methods in simulations and a robotic implementation.

The second problem that we considered is the team pursuit which is closely related to the assignment problem. A heterogeneous team of robotic agents is given the task of pursuing a heterogeneous team of moving target positions. In its core, this is a target assignment problem with a set of moving targets. By assigning the agents that are capable of moving at higher velocities to faster targets, an optimal pursuit was achieved. A decentralized algorithm for finding the optimal assignment was introduced. Through the use of a stochastic extension based on the Metropolis-Hastings algorithm, the requirements on the communication graph could be reduced to the natural requirement of connectivity. We discussed different choices for pursuit performance measures leading to different optimal assignments. An interesting observation is the tendency to abandon single targets if this helps the rest of the team for some performance measures. Again, we investigated the optimal team composition when introducing a velocity cost. This was motivated by practical considerations and substantiated with a real-world example where a budget for retail part cost was used as a

constraint on the team composition. The methods have been validated in both simulation and implementation.

Both of these problems showed the advantages and limitations of slow robots when incorporated into a team. A unifying theme was finding a compromise between task performance and energy consumption when assembling a team of agents for a specific task. With this thesis we have also shifted the perspective on solving tasks with a multi-robot team from pure control design towards the choice of the team itself.

5.1 Outlook

Interesting considerations that were beyond the scope of this thesis include:

- Applying the methods to systems with more complex heterogeneous dynamics. Different locomotion within the team can enable some agents to move to places that others can not reach or reduce the time required to get there. This would be an interesting topic especially with regards to the pursuit problem, where different agent locomotion can be more or less suitable for the pursuit of a target with known dynamics.

- Introducing a form of competition between the agents and their opponents. Through co-evolution, the effects of heterogeneity on fitness in a competitive setting could be evaluated.

- Combining temporal heterogeneity with other kinds of heterogeneity. In many heterogeneous teams, temporal heterogeneity would come as an unavoidable side effect of having a team with different skill sets. Investigating this connection could lead to insights about the interplay of skills and speed.

- Investigating different team sizes. We have only investigated teams of a given fixed size. Instead, one could also do a comparison in numbers, replacing a single fast robot by a whole group of slower agents. Depending on the task, a fast robot might be outperformed by just a small number of slow robots or it might be more suitable than even a large team of slow robots.

REFERENCES

[1] Altieri, M., *Biodiversity and Pest Management in Agroecosystems*. Food Products Press, 1994.

[2] Arkin, R. C. and Egerstedt, M., "Temporal heterogeneity and the value of slowness in robotic systems.," in *IEEE Conference on Robotics and Biomimetics*, 2015.

[3] Bailey, T., Bryson, M., Mu, H., Vial, J., McCalman, L., and Durrant-Whyte, H. F., "Decentralised cooperative localisation for heterogeneous teams of mobile robots.," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2859–2865, May 2011.

[4] Becchi, M. and Crowley, P., "Dynamic thread assignment on heterogeneous multiprocessor architectures," in *Proceedings of the 3rd Conference on Computing Frontiers*, CF '06, (New York, NY, USA), pp. 29–40, ACM, 2006.

[5] Bertsekas, D. P., "The auction algorithm: A distributed relaxation method for the assignment problem," *Annals of Operations Research*, vol. 14, no. 1, pp. 105–123, 1988.

[6] Borrmann, U., Wang, L., Ames, A. D., and Egerstedt, M., "Control barrier certificates for safe swarm behavior," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 68 – 73, 2015.

[7] Britannica, E., "Agressive mimicry," in *Encyclopædia Britannica Online*, Encyclopædia Britannica, 2016. Accessed May 27, 2016.

[8] Buchstab, A., *KUKA: Innovations in Industrial Robotics*, pp. 267–274. Vienna: Springer Vienna, 2013.

[9] Bullo, F., Cortes, J., and Martinez, S., *Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms*. Princeton, NJ, USA: Princeton University Press, 2009.

[10] Cardaliaguet, P., Quincampoix, M., and Saint-Pierre, P., "Pursuit differential games with state constraints," *SIAM Journal on Control and Optimization*, vol. 39, no. 5, pp. 1615–1632, 2000.

[11] Chaimowicz, L., Grocholsky, B., Keller, J. F., Kumar, V., and Taylor, C. J., "Experiments in multirobot air-ground coordination," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 4, pp. 4053–4058 Vol.4, April 2004.

[12] Cortes, J., Martinez, S., Karatas, T., and Bullo, F., "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, pp. 243–255, April 2004.

[13] Croucher, N. J., Coupland, P. G., Stevenson, A. E., Callendrello, A., Bentley, S. D., and Hanage, W. P., "Diversification of bacterial genome content through distinct mechanisms over different timescales," *Nature Communications*, vol. 5, Nov. 2014.

[14] Dellaert, F., Balch, T. R., Kaess, M., Ravichandran, R., Alegre, F., Berhault, M., McGuire, R., Merrill, E., Moshkina, L. V., and Walker, D., "The georgia tech yellow jackets: A marsupial team for urban search and rescue.," *AAAI Mobile Robot Competition*, vol. 2002, pp. 44–49, 2002.

[15] Diaz-Mercado, Y., Lee, S. G., and Egerstedt, M., "Distributed dynamic density coverage for human-swarm interactions," in *2015 American Control Conference (ACC)*, pp. 353–358, July 2015.

[16] Dudek, G., Jenkin, M. R. M., Milios, E., and Wilkes, D., "A taxonomy for multi-agent robotics," *Autonomous Robots*, vol. 3, no. 4, pp. 375–397, 1996.

[17] Elfes, A., Bergerman, M., Carvalho, J. R. H., de Paiva, E. C., Ramaos, J., and Bueno, S. S., "Air-ground robotic ensembles for cooperative applications: Concepts and preliminary results," in *2nd International Conference on Field and service Robotics, Pittsburgh, Pa (USA)*, pp. 75–80, 1999.

[18] Flynn, J., "Lion and man: A bounded pursuit game," Tech. Rep. 161, Department of Statistics, Stanford University, Stanford, CA, oct 1970.

[19] Flynn, J., "Lion and man: The general case," *SIAM Journal on Control*, vol. 12, no. 4, pp. 581–597, 1974.

[20] Franconeri, S. L. and Simons, D. J., "Moving and looming stimuli capture attention," *Perception & Psychophysics*, vol. 65, pp. 999–1010, Oct. 2003.

[21] Gonzalez-Rodriguez, D. and Hernandez-Carrion, J. R., "A bacterial-based algorithm to simulate complex adaptive systems," in *From Animals to Animats 13: 13th International Conference on Simulation of Adaptive Behavior. Proceedings* (Pobil, A. P., Chinellato, E., Martinez-Martin, E., Hallam, J., Cervera, E., and Morales, A., eds.), pp. 250–259, Springer International Publishing, July 2014.

[22] Grabowski, R., Navarro-Serment, L. E., Paredis, C. J., and Khosla, P. K., "Heterogeneous teams of modular robots for mapping and exploration," *Autonomous Robots*, vol. 8, no. 3, pp. 293–308, 2000.

[23] Grocholsky, B., Keller, J., Kumar, V., and Pappas, G., "Cooperative air and ground surveillance," *IEEE Robotics Automation Magazine*, vol. 13, pp. 16–25, Sept 2006.

[24] Hastings, W. K., "Monte carlo sampling methods using markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.

[25] Hens, L. and Boon, E. K., "Causes of biodiversity loss: a human ecological analysis," *O Futuro dos Recursos*, vol. 1, Oct. 2003.

[26] Hourani, H., Wolters, P., Hauck, E., and Jeschke, S., *Intelligent Robotics and Applications: 4th International Conference, ICIRA 2011, Aachen, Germany, December 6-8, 2011, Proceedings, Part I*, ch. A Marsupial Relationship in Robotics: A Survey, pp. 335–345. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[27] Howard, A., Parker, L. E., and Sukhatme, G. S., "Experiments with a large heterogeneous mobile robot team: Exploration, mapping, deployment and detection," *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 431–447, 2006.

[28] Jiang, Z.-P. and Nijmeijer, H., "Tracking control of mobile robots: A case study in backstepping," *Automatica*, vol. 33, no. 7, pp. 1393 – 1399, 1997.

[29] Jongepier, E. and Foitzik, S., "Fitness costs of worker specialization for ant societies," *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 283, no. 1822, 2016.

[30] Khoury, C. K., Bjorkman, A. D., Dempewolf, H., Ramirez-Villegas, J., Guarino, L., Jarvis, A., Rieseberg, L. H., and Struik, P. C., "Increasing homogeneity in global food supplies and the implications for food security," *Proceedings of the National Academy of Sciences*, vol. 111, no. 11, pp. 4001–4006, 2014.

[31] Knuth, D. E., *The Art of Computer Programming, Vol. 3: Sorting and Searching*. Redwood City, CA, USA: Addison Wesley Longman Publishing, second ed., 1998.

[32] Lee, S. G., Diaz-Mercado, Y., and Egerstedt, M., "Multirobot control using time-varying density functions," *IEEE Transactions on Robotics*, vol. 31, pp. 489–493, April 2015.

[33] Lee, S.-O., Cho, Y.-J., Hwang-Bo, M., You, B.-J., and Oh, S.-R., "A stable target-tracking control for unicycle mobile robots," in *International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, pp. 1822–1827, 2000.

[34] Lee, T.-C., Song, K.-T., Lee, C.-H., and Teng, C.-C., "Tracking control of unicycle-modeled mobile robots using a saturation feedback controller," *IEEE Transactions on Control Systems Technology*, vol. 9, pp. 305–318, Mar 2001.

[35] Lewin, J., "The lion and man problem revisited," *Journal of Optimization Theory and Applications*, vol. 49, no. 3, pp. 411–430, 1986.

[36] Littlewood, J. E. and Bollobás, B., *Littlewood's miscellany*. Cambridge University Press, 1986.

[37] Mesbahi, M. and Egerstedt, M., *Graph Theoretic Methods in Multiagent Networks*. Princeton series in applied mathematics, Princeton (N.J.): Princeton University Press, 2010.

[38] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E., "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, 1953.

[39] Michael, N., Zavlanos, M. M., Kumar, V., and Pappas, G. J., "Distributed multi-robot task assignment and formation control," in *IEEE International Conference on Robotics and Automation*, pp. 128–133, May 2008.

[40] Mihai, A., "The fastest cars in the world, decade by decade." http://www.zmescience.com/research/technology/fastest-cars/, Aug 2015. [Online; accessed 08-June-2016].

[41] Munkres, J., "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[42] Nyakatura, J. A. and Fischer, M. S., "Functional morphology of the muscular sling at the pectoral girdle in tree sloths: convergent morphological solutions to new functional demands?," *Journal of Anatomy*, vol. 219, no. 3, pp. 360–374, 2011.

[43] Pauli, J. N., Mendoza, J. E., Steffan, S. A., Carey, C. C., Weimer, P. J., and Peery, M. Z., "A syndrome of mutualism reinforces the lifestyle of a sloth," *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 281, no. 1778, 2014.

[44] Pickem, D., Egerstedt, M., and Shamma, J. S., "A game-theoretic formulation of the homogeneous self-reconfiguration problem," *CoRR*, 2015.

[45] Pickem, D., Wang, L., Glotfelter, P., Diaz-Mercado, Y., Mote, M., Ames, A., Feron, E., and Egerstedt, M., "Safe, remote-access swarm robotics research on the Robotarium," *CoRR*, vol. abs/1604.00640, 2016.

[46] Potter, M. A., Meeden, L. A., Schultz, A. C., and Intelligence, A., "Heterogeneity in the coevolved behaviors of mobile robots: The emergence of specialists," in *International joint conference on artificial intelligence*, vol. 17, pp. 1337–1343, 2001.

[47] Reed, D. H. and Frankham, R., "Correlation between fitness and genetic diversity," *Conservation Biology*, vol. 17, no. 1, pp. 230–237, 2003.

[48] Ren, W. and Beard, R. W., *Distributed Consensus in Multi-vehicle Cooperative Control*. London: Springer-Verlag, 2008.

[49] Rybski, P. E., Stoeter, S. A., Erickson, M. D., Gini, M., Hougen, D. F., and Papanikolopoulos, N., "A team of robotic agents for surveillance," in *Proceedings of the Fourth International Conference on Autonomous Agents*, AGENTS '00, (New York, NY, USA), pp. 9–16, ACM, 2000.

[50] Smit, L. T., Smit, G. J., Hurink, J. L., Broersma, H., Paulusma, D., and Wolkotte, P. T., "Run-time assignment of tasks to multiple heterogeneous processors," in *5th PROGRESS Symposium on Embedded Systems*, pp. 185–192, STW Technology Foundation, 2004.

[51] Spall, J. C., *Introduction to Stochastic Search and Optimization*. John Wiley & Sons, Inc., 2005.

[52] Sukhatme, G., Montgomery, J. F., and Vaughan, R. T., "Experiments with cooperative aerial-ground robots," *Robot Teams: From Diversity to Polymorphism*, pp. 345–368, 2001.

[53] Tang, C. P., "Differential flatness-based kinematic and dynamic control of a differentially driven wheeled mobile robot," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 2267–2272, Dec 2009.

[54] Tarín, C., Brugger, H., Tibken, B., and Hofer, E. P., "Globally asymptotically stable position control for an autonomous mobile robot," in *European Control Conference (ECC)*, pp. 268–271, Aug 1999.

[55] Troscianko, T., Benton, C. P., Lovell, P. G., Tolhurst, D. J., and Pizlo, Z., "Camouflage and visual perception," *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, vol. 364, no. 1516, pp. 449–461, 2009.

[56] Venkat, A. and Tullsen, D. M., "Harnessing isa diversity: Design of a heterogeneous-isa chip multiprocessor," in *Proceeding of the 41st Annual International Symposium on Computer Architecuture*, ISCA '14, (Piscataway, NJ, USA), pp. 121–132, IEEE Press, 2014.

[57] Wang, X., Li, M., Wang, P., Guo, W., and Sun, L., "Bio-inspired controller for a robot cheetah with a neural mechanism controlling leg muscles," *Journal of Bionic Engineering*, vol. 9, no. 3, pp. 282 – 293, 2012.

[58] Wilson, E. O., "Caste and division of labor in leaf-cutter ants (hymenoptera: Formicidae: Atta)," *Behavioral Ecology and Sociobiology*, vol. 7, pp. 157–165, July 1980.

[59] Wurm, K. M., Dornhege, C., Nebel, B., Burgard, W., and Stachniss, C., "Coordinating heterogeneous teams of robots using temporal symbolic planning," *Autonomous Robots*, vol. 34, no. 4, pp. 277–294, 2013.

[60] Young, H. P., "The evolution of conventions," *Econometrica*, vol. 61, pp. 57–84, January 1993.