MULTI-OBJECTIVE CONTROL AND COORDINATION OF MULTI-AGENT SYSTEMS

BY

CHRISTOPHER GREGORY VALICKA

DISsertATION

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Systems & Entrepreneurial Engineering in the Graduate College of the University of Illinois at Urbana-Champaign, 2013

Urbana, Illinois

Doctoral Committee:

Associate Professor Dušan M. Stipanović, Chair
Associate Professor Ali Abbas
Associate Professor Ramavarapu Sreenivas
Professor Petros Voulgaris
Abstract

In this dissertation, control methodologies for systems of multiple mobile agents facing multiple objectives are considered. Carefully constructed objective functions for agents interested in any or all of collision avoidance, robust communications, waypoint following, and dynamic coverage objectives are presented. Theory relevant to gradient-like control laws designed using over-approximations of the maximum function is studied and applications of developed controllers in simulations and physical experiments are presented.

Agent behavior resulting from and relevant to the various objectives is discussed. Amongst the various objective functions, novel formulations for proximity and global coverage objectives are presented. A novel control law design is developed based on copula structures from multiattribute decision theory. Systematic and intuitive methods for determining the tradeoffs between multiple objectives are presented and applied to the design of multi-agent control laws.

Using over-approximations of the maximum function, appropriate definitions are presented to develop the conditions required for an agent to accomplish multiple objectives. The proposed control laws’ suitability for and ability to be implemented straightforwardly on various multi-agent systems facing multiple objectives are demonstrated in multiple simulation and experimental results. The proposed copula based control law and global coverage approach are discussed and validated through their combined implementation in a multi-robot testbed.
To my wife, for embracing our adventure.
Acknowledgments

The challenges, the highs and lows, the learning and growing that I have experienced while producing the work contained herein were endured, persevered, cultured, motivated, and encouraged thanks in large part to several instrumental people. Unquestionably, the majority of my gratitude is for Dušan. Throughout my time at the University of Illinois at Urbana-Champaign, Dušan has remained an immutable and selfless mentor and friend. Dušan has equipped me with the tools necessary to explore scientific details and the sensibility to put the math and engineering in perspective. A special thanks is due to Professor Abbas for his inspiring course and his professional and intuitive approach to engineering problems. Thank you, too, Professor Sreenivas for the opportunity and encouragement to explore the applications of engineering problems in the field of computer science, something I very much enjoy. A thank you is also deserved by Professor Voulgaris for his courses, flexibility, and conversations, technical and otherwise.

Dan Block, for all the opportunities, knowledge, and experience you have given me, thank you. Your instruction, patience, and willingness to share wisdom, camaraderie, and robots are priceless. Working for you and with you has been an honor and privilege. A big thank you to Rick Rekoske too for the countless number of hours spent implementing, developing, proofreading, and explaining everything related to the multi-robot testbed. Also, a thank you to Mark Michelotti for being a great friend and for developing Linux code for the robots with me.

A special thank you to Gokhan Atinc and Mika Panagou, for your friendship and multitude of technical conversations and for being outstanding academic mentors. Thank you Carlos for a wonderful time spent exploring coverage control and helping me practice my Spanish. Of course, thank you too, Anshuman Mishra for your wealth of knowledge on all things technical and for being a selfless friend.
Table of Contents

List of Figures .......................................................... vii

Chapter 1 Introduction .................................................. 1
  1.1 Outline ............................................................. 2
  1.2 Background ....................................................... 3
  1.3 Summary of Contributions ...................................... 5

Chapter 2 Objective Functions ........................................ 8
  2.1 Waypoint Following ............................................... 8
  2.2 Collision Avoidance .............................................. 9
  2.3 Proximity and Flocking .......................................... 12
  2.4 Dynamic Coverage ............................................... 14
  2.5 Dynamic Global Coverage Coordination ......................... 18

Chapter 3 Multiattribute Copulas and Utility Functions ............ 21
  3.1 Multiattribute Utility Functions .............................. 21
  3.2 Copulas ................................................................ 22
  3.3 $n$-Dimensional Copulas ......................................... 23
  3.4 Multiattribute Utility Copulas .................................. 25

Chapter 4 Control Design Using Approximations to the Maximum 
  Function ................................................................. 28
  4.1 Maximum Approximation With $p$-Norms ..................... 29
  4.2 Maximum approximation With Multiattribute Utility Copulas 31
  4.3 Control Using Goal Functions .................................. 34
  4.4 Particulars of Goal Function Controllers ........................ 37
  4.5 Discussion of Goal Functions .................................... 39

Chapter 5 Simulations and Experiments ................................ 40
  5.1 MATLAB Multi-Agent Simulator ................................ 41
  5.2 Multi-Robot Testbed .............................................. 42
  5.3 Initial Simulation Development ................................. 46
  5.4 $p$-norm Goal Function .......................................... 49
  5.5 Multiattribute Utility Copula Controller ...................... 56
  5.6 Global Coverage Objective ...................................... 61
  5.7 Experiments With Copula-Based Goal Function .............. 66
List of Figures

2.1 Sensor profile .............................................. 15
5.1 Example plots produced by MATLAB multi-agent simulator . 42
5.2 Multi-robot testbed ....................................... 43
5.3 Example shared information flow of multi-robot testbed .... 43
5.4 Motion capture software and camera setup .................. 44
5.5 Robot with markers and electronics suite .................... 44
5.6 Screenshot of testbed visualization software ............... 45
5.7 Agent trajectories within the coverage domain ............. 46
5.8 Level of coverage and final agent positions over coverage domain 48
5.9 Global coverage error with respect to time ................ 48
5.10 Pairwise agent and agent to obstacle distances .......... 49
5.11 Level of coverage and final agent positions over coverage domain 50
5.12 Agent trajectories with respect to the coverage domain ... 50
5.13 Global coverage error with respect to time ............... 52
5.14 Pairwise agent and agent to obstacle distances .......... 52
5.15 Initial robot positions and distances .................... 53
5.16 Intermediate robot positions and distances ............... 54
5.17 Final robot positions and distances ..................... 54
5.18 Agent trajectories with waypoints and obstacles .......... 57
5.19 Inter-agent and agent-waypoint and agent-obstacle distances . 58
5.20 $U(w_{11}^o, \bar{w}_{11}^*) = 0.95, U(w_{12}^o, \bar{w}_{12}^*) = 0.01, \epsilon = 0.001$ .......... 59
5.21 $U(w_{11}^o, \bar{w}_{11}^*) = 0.1, U(w_{12}^o, \bar{w}_{12}^*) = 0.7, \epsilon = 0.001$ .......... 60
5.22 Single integrator agent trajectories ........................ 62
5.23 Nonholonomic agent trajectories ............................ 63
5.24 Pairwise agent and agent-obstacle distances, single integrator models ........................................ 64
5.25 Pairwise agent and agent-obstacle distances, nonholonomic models ............................................. 64
5.26 Coverage error versus time of single integrator agent 1 .... 65
5.27 Coverage error versus time of nonholonomic agent 1 ....... 66
5.28 Single integrator agents’ goal functions versus time ....... 67
5.29 Nonholonomic agents’ goal functions versus time ......... 67
5.30 Final single integrator agent configuration ................ 68
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.31 Final nonholonomic agent configuration</td>
<td>68</td>
</tr>
<tr>
<td>5.32 Robot trajectories for four robot experiment</td>
<td>69</td>
</tr>
<tr>
<td>5.33 Robot trajectories for three robot experiment</td>
<td>70</td>
</tr>
<tr>
<td>5.34 Pairwise robot distances</td>
<td>71</td>
</tr>
<tr>
<td>5.35 Pairwise robot distances</td>
<td>71</td>
</tr>
<tr>
<td>5.36 Pairwise robot-obstacle distances</td>
<td>72</td>
</tr>
<tr>
<td>5.37 Pairwise robot-obstacle distances</td>
<td>73</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Unmanned and autonomous systems continue to be an area of fervent research and industry focus as more and more they represent a cost effective manner to accomplish hazardous and complex tasks with reduced human supervision. Through advances in the reliability and cost of computing and communication capabilities, simple and inexpensive yet powerful subsystems can be combined to produce large scale systems capable of achieving complex goals. By using well defined objectives and constraints, large-scale systems comprised of many subsystems, often referred to as multi-agent systems, can accomplish complex goals often times impossible for smaller systems. According to the modeling of the objectives and capabilities of the agents, control laws can be designed in a centralized or distributed manner to produce desired multi-agent system behavior and often times new, emergent behaviors.

Challenges for multi-agent system control law design arise when various objectives or constraints conflict. These challenges may depend on the dynamical models used to describe the agents. In general, the set of behaviors achievable for systems modeled with linear dynamics differs from the set of achievable behaviors for agents modeled with nonlinear dynamics. Multi-agent systems may be comprised of homogenous or heterogenous subsystems, necessitating particular considerations for constraints, objectives, or laws of interaction. Differences between agents can have considerable influence on objectives such as safety verification, reliable communication, coordinated movement, or other common goals such as waypoint following, trajectory tracking, and dynamic coverage.

The research presented herein aims to address several of these challenges and through simulation and implementation on physical systems, validate approaches considered for heterogenous systems modeled within a broad class of dynamical systems. Current literature related to the control of multi-
agent systems considers various control law designs and objective function definitions.

1.1 Outline

The dissertation is organized into six chapters. In Chapter 2, desired behavior, interaction laws, and constraints are formulated in terms of objective functions for multi-agent systems intending to accomplish multiple objectives. The focus of this chapter is on defining the objectives based on capabilities common to modern autonomous systems whether they be land, sea, air, or space based. The set of objectives considered herein represent a subset of those generally encountered by autonomous systems and consists of essential and commonly desired objectives. In general, they take on a non-negative definite formulation which make them suitable for Lyapunov-like analysis. Amongst the essential objectives, trajectory based laws and interactions are discussed through waypoint following, dynamic coverage, and global coverage formulations. Multi-agent system missions in which these objectives are relevant are presented. The objective of safety verification is encapsulated in terms of remaining collision free of other agents and static obstacles. Commonly desired behaviors of flocking and maintaining reliable communications is described through various distance metrics, or proximity, based objective functions. Chapter 3 presents preliminaries related to multiattribute utility copulas. Theory related to the properties of copulas and multiattribute utility copulas is presented and a formal method for assigning preferences amongst the multiple objectives is shown. The preliminaries are presented towards the end of using multiattribute utility copulas as goal functions, or multi-objective scalarizations. Chapter 4 discusses control law design using goal functions constructed as approximations to the maximum function. Goal function formulations constructed using $p$-norms and multiattribute utility copulas are presented and formal relationships between the formulations and an over-approximation of the maximum function are demonstrated. Furthermore, a definition is given to precisely define the relationship between all objectives being satisfied and a goal being accomplished. The topic of accomplishing all objectives through use of control laws defined in terms of a goal function is discussed. The class of agent dynamic mod-
els for which the considered control law approaches are suitable is presented along with specific examples. Details related to the use of gradient-like control laws based on goal functions are presented. The use of $p$-norm and multiattribute utility copula control laws are discussed with attention given to determining the tradeoffs between multiple objectives. Chapter 5 presents simulation and experimental results from implementing goal function based control laws on multi-agent systems. Systems defined using both single integrator and nonholonomic dynamic models are simulated and experiments are conducted using car-like robots modeled as nonholonomic systems. In the presented simulations and experiments, a variety of systems sizes, in terms of agents, and of objectives are considered. Chapter 6 presents conclusions of the research conducted to date as well as directions and areas for future research.

1.2 Background

Solving multi-objective optimization and control problems is particularly challenging when the constraints, or alternatively the objectives, conflict with each other. The challenges of these problems increase if multiple decision makers are included or if the decision makers are dynamic systems modeled using differential equations. In order to avoid difficulties associated with vector optimizations, multi-objective problems are often solved after constructing a scalarization of the objective functions [1]. Some examples in which a system of multiple decision makers intend to solve a multi-objective problem come from the field of differential games. In [2, 3], the agents use approximations to the minimum and maximum functions to scalarize their objective functions. By using differential inequalities and Lyapunov control functions [4, 5], multiple objectives are accomplished simultaneously in [6].

According to the nature of feasible solutions, multi-objective problems may benefit more from the tools derived from either multi-objective optimization or multiattribute decision analysis [1]. In the field of multiattribute decision analysis, decisions are commonly parameterized using a specific set of descriptive attributes. In decisions of a deterministic nature, decision makers face the important task of constructing a value function that appropriately captures tradeoffs amongst attributes. In decision situations including un-
certain outcomes, utility functions can be assigned with respect to the value function to accurately reflect decision makers’ attitudes toward the set of possible outcomes [7, 8, 9]. Copula structures can be used to construct multiattribute utility functions from combinations of individual utility functions [10] and can be used to incorporate many of the common forms of attribute independence [10, 11].

A class of problems benefitting from the approaches of multi-objective optimization and multiattribute decision analysis is control and coordination of multi-agent systems intending to accomplish multiple objectives. Research into applications of various methodologies which address these problems constitute an active area of development [12, 13]. Several alternative approaches to multi-agent control and coordination include the work of [14, 15, 16, 17]. Commonly, alternative approaches use potential field methods as in [18, 19, 20, 21]. Examples of control law and coordination implementations on multi-robot testbeds can be found in [22, 23, 24, 25, 26, 27, 28, 29]. A multi-robot testbed in which coverage objectives are considered is presented in [30]. An example of using multiattribute utility functions for control and coordination of multi-agent systems presented herein was originally demonstrated in [31].

Also relevant to the problems of control and coordination of multi-agent systems is the set of considered objectives and their associated definitions. Collision avoidance is essential to multi-agent systems intending to accomplish any other objective. The focus herein considers the formulation originally established in [32] based on work in the area of avoidance control developed in [33, 34, 35]. Formation or proximity based constraints are also particularly relevant to multi-agent systems and represent a broad area of research. See, for example, [36, 37] which are examples of leader-follower arrangements. A detailed analysis of a broad class of flocking behaviors is presented in [38]. Objective functions herein instead consider a simplified form presented in [39, 40] with the purpose of modeling communication constraints. A focus within this work is also given to the objective of dynamic coverage. Previous work related to single integrator systems was considered in [39, 41, 42] which used various methods and assumptions for achieving global coverage and [43] which studied problems where coverage information decays. The definition of dynamic coverage considered herein follows the definition given in [39, 40] and differs from the coverage control problem for-
mulation considered in [15] which studies the deployment of mobile sensors and the persistence of coverage problem considered in [44, 45].

1.3 Summary of Contributions

Multi-objective control and coordination of multi-agent systems represents a broad class of problems. By combining the well established approaches of multi-objective optimization and multiattribute decision analysis, this dissertation presents new tools for designing control laws for multi-agent systems tasked with multiple objectives. Systematic and intuitive methods from multiattribute decision analysis are adapted for determining the tradeoffs between the agents’ respective objectives. Specific implementation examples from simulations and experiments provide an insight into the effectiveness of the control law designs for heterogenous multi-agent systems of differing sizes, with different dynamic models, and tasked with various objectives. By making use of the control laws presented herein to combine multiple objectives, complex behavior that accomplishes all objectives can be tuned to fit the decision maker’s preferences.

Chapter 2 presents a rich set of nonnegative definite objective functions to model behaviors and constraints related to waypoint following, collision avoidance, proximity control, and dynamic coverage. A novel proximity objective function formulation, analogous to the definition of the collision avoidance objective function, is presented. Specific details of the coverage objective are presented related to singular control and a behavior, discovered through simulation, resulting from the coverage formulation. A novel method for ensuring that agents satisfactorily cover points in the coverage domain even after entering local minima is presented. By making use of a simple search algorithm and a careful definition of an agent’s nearest uncovered point, an effective global coverage scheme is incorporated without the need to design an additional switching control law. Behaviors resulting from and intended by the various objective function definitions are discussed.

In Chapter 3, details and preliminaries of multiattribute utility theory are presented towards the end goal of adapting a tool originally used in multiattribute decision analysis for use in multi-objective control. A review of tools originally developed to construct joint probability functions from
marginal probability functions is conducted. Formal proofs are given that establish a relationship between two-dimensional copulas, and then for general $n$-dimensional copulas, and an approximation of the minimum function from below. Explicit formulations for a Class 1 multiattribute utility copula and multiplicative generating function are given. An intuitive and systematic process, indifference probability assessments, to determine the tradeoffs among a set of $n$ attributes is described in detail.

Chapter 4 studies the design of control laws for multi-objective control problems through the use of goal functions. Theoretical results from using $p$-norms and multiattribute utility copulas as approximations to the maximum function from above are presented. The relationship between both approximations and the maximum function is explicitly demonstrated. Formal definitions of goal functions and of accomplishing all objectives within a goal function are rigorously established. This is accomplished by using the comparison principle, differential inequalities and Lyapunov-like analysis. Results are presented appropriate for a general class of dynamic systems affine in control. Details related to using gradient-like control laws to achieve the Lyapunov-like analysis are discussed and several control law designs are presented. This includes instances when agent dynamics are modeled as single integrator systems as well as instances when agent dynamics are modeled as nonholonomic systems. Key implementation related features of both goal function formulations are also discussed. Ultimately, a variety of explicit, gradient-like control laws designed to accomplish multiple objectives simultaneously are presented.

Finally, Chapter 5 presents the results of implementing the control law designs of Chapter 4 on a number of different multi-agent systems. An extensible MATLAB simulator and multi-robot testbed are developed. Within the implementations presented, first considered are multi-agent systems modeled with single integrator dynamics controlled using control laws not based on goal functions. A more complex simulation involving $p$-norm based goal functions and agent dynamics modeled as nonholonomic systems is then presented toward the goal of implementing the control laws on car-like mobile robots. Both simulations and experiments in which all agents accomplish all of their respective objectives are presented. A variety of complex behaviors are produced through the use of heterogenous objective function parameters, controller gains, and differing sets of assigned objectives. These behaviors
are depicted in numerous figures displaying data from the various simulations and experiments. The ability of goal functions designed using multiattribute utility copulas to streamline the selection of attribute tradeoff parameters is demonstrated through their use on both single integrator and nonholonomic agents as well as on a multi-robot testbed. The ability to incorporate additional objectives is demonstrated by the use of the global coverage objective in both simulations and experiments. The ability of the new global coverage scheme to accomplish global coverage and its ability to be implemented in a straightforward manner are demonstrated. Ultimately, a framework for creating complex yet achievable behavior through the use of well-defined objective functions and goal functions is validated in various simulations and experiments.
Chapter 2

Objective Functions

The purpose of this chapter is to present the set of objectives and corresponding objective function formulations considered for the purpose of designing control laws for multi-agent systems. Behaviors related to collision avoidance, flocking or proximity based maintenance of robust communications, waypoint following, dynamic coverage, and global coverage will be discussed and encapsulated in various nonnegative definite objective functions. Collision avoidance behavior follows from a formulation originally presented in [32], and serves as an inspiration for a flocking objective function presented later. An additional proximity objective function will be presented, its form and the formulations for dynamic coverage follow from those used in [39, 40]. A new global coverage objective formulation will be presented along with an algorithm used to determine an agent’s nearest uncovered point. By combining the objectives considered herein, different simple or complex multi-agent system behaviors can be achieved through the use of control laws presented in Chapter 4.

2.1 Waypoint Following

Waypoint following is an objective that is commonly used in the control and coordination of semi-autonomous and autonomous multi-agent systems. By ordering a set of waypoints and assigning a new waypoint once an agent has arrived at the first waypoint in the list, waypoint trajectories can be created. While not used extensively in the simulations and experiments considered herein, this objective is worth discussing as it fits within the theme of nonnegative definite objective functions as a simple example. Distance between the position of an agent and its waypoint represent a natural metric for multiple objective problems which include waypoint following. Let the waypoint
following objective function \( v_{\text{wt}}^i \) be defined as follows:

\[
v_{\text{wt}}^i(x) := \|x_i - x_{\text{des}}^i\|^2, \quad v_{\text{wt}}^i \in [0, \infty),
\]

where \( \| \cdot \| \) is the standard Euclidean norm and \( x_{\text{des}}^i \) represents the desired position of agent \( i \). Minimizing the objective function \( v_{\text{wt}}^i \) along its gradient to a value less than some threshold \( \epsilon \), agent \( i \)’s position, denoted by \( x_i \), can be made arbitrarily close to the desired waypoint position of agent \( i \), \( x_{\text{des}}^i \).

2.2 Collision Avoidance

Although considered a secondary objective to the more common objectives of trajectory tracking, waypoint following, or coverage control (to be discussed later in this chapter), collision avoidance is of paramount importance to the safety and usability of multi-agent autonomous systems. In comparison to approaches for guaranteed collision avoidance which rely on a path planning or proactive scheme, the objective functions considered herein are reactive in nature. That is to say, outside of bounded regions surrounding the agents, no control effort calculations specific to collision avoidance are made.

It is worthwhile for the control designer of multi-agent systems to consider the collision avoidance objective from two perspectives. Namely, scenarios when the obstacles will cooperatively avoid the agents and scenarios in which obstacles are noncooperative. The scope of this paper considers collision avoidance from both perspectives, with inter-agent collision avoidance considered from a cooperative perspective and static obstacle avoidance from the noncooperative perspective. Noncooperative scenarios in which the agents are required to avoid obstacles capable of pursuing the agents are not considered.

2.2.1 Inter-agent Collision Avoidance

Following the methodology and the objective function given in [32], the collision avoidance objective function between cooperating agents \( i \) and \( j \) is
defined as follows:

\[ v_{ij}^a(x) := \left( \min \left\{ 0, \frac{\|x_i - x_j\|^2_{P_{ij}} - R_{ij}^2}{\|x_i - x_j\|^2_{P_{ij}} - r_{ij}^2} \right\} \right)^2. \] (2.2)

Here, \( R_{ij} \) and \( r_{ij} \) represent the detection and avoidance radii, respectively, of agent \( i \) with respect to agent \( j \). The detection and avoidance radii are required to be nonzero and satisfy \( R_{ij} > r_{ij} \). The radii are design parameters selected by considering agent \( i \)'s shape and size, sensing capabilities, and/or locomotive power in relation to agent \( j \). The sensing capabilities of agent \( i \) are of particular importance when selecting both the detection and avoidance radii. Defining the design parameters pairwise allows for the flexibility of incorporating the aforementioned characteristics of non-homogenous multi-agent systems. In (2.2), agent \( i \)'s and \( j \)'s positions are denoted by \( x_i \) and \( x_j \), respectively. \( P_{ij} \) is a positive definite matrix which can be used to elongate the circular detection and avoidance regions of agent \( i \) with respect to agent \( j \).

In a multi-agent system, the objective function of avoiding of \( N \) cooperating agents can be formulated using the sum of agent \( i \)'s respective avoidance functions:

\[ \nu_i^a = \sum_{j=1, j \neq i}^{N} v_{ij}^a, \ j = 1, \ldots, N. \] (2.3)

### 2.2.2 Static Obstacle Avoidance

The objective function encapsulating collision avoidance between agent \( i \) and static obstacle \( k \) takes on the same form as (2.2)

\[ v_{ik}^a(x) := \left( \min \left\{ 0, \frac{\|x_i - x_k\|^2_{P_{ik}} - R_{ik}^2}{\|x_i - x_k\|^2_{P_{ik}} - r_{ik}^2} \right\} \right)^2. \] (2.4)

As before, \( R_{ik} > r_{ik} \), with \( r_{ik} \) strictly positive, and \( P_{ik} \) is positive definite.

Although the objective function remains unchanged between the scenarios of cooperating agents and static obstacles, it may be in the interest of the control designer to consider the parameters of (2.4) in terms of the static obstacle to be avoided. These parameters need to be selected such that \( \|x_i - x_k\|^2_{P_{ik}} \leq r_{ik}^2 \) over-bounds obstacle \( k \). A convenient way to ensure this
and to allow for arbitrary orientation of two-dimensional ellipsoidal regions is by using the following equation:

\[ P_{ik} = R^T_\theta(\theta) \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix} R_\theta(\theta), \ 0 < \alpha, \beta \leq 1 \]  

(2.5)

where \( R_\theta(\theta) \) is the counterclockwise rotation matrix of the x and y axes, respectively extended by \( \frac{1}{\alpha} \) and \( \frac{1}{\beta} \), of the ellipse. The combined collision avoidance function for agent \( i \) in a scenario with \( N_k \) static obstacle then becomes

\[ v^a_i = \sum_{j=1}^{N} v^a_{ij} + \sum_{k=1}^{N_k} v^a_{ik}, \ j \neq i, \ j = 1, \ldots, N, \ k = N + 1, \ldots, N + N_k. \]  

(2.6)

The reactive nature of the avoidance objective function in (2.2) makes it a desirable formulation of the collision avoidance function for the control designer of a multi-agent system. Additionally, this formulation lends itself nicely to Lyapunov analysis when designing gradient based controllers and attempting to establish guaranteed collision avoidance. This is evidenced by noting that (2.2) is nonnegative definite with a nonpositive definite gradient

\[
\frac{\partial v^a_{ij}}{\partial x_i} = \begin{cases} 
0 & \text{if } \|x_i - x_j\|_P \geq R_{ij} \\
\frac{4(R_{ij}^2 - r_{ij}^2)(\|x_i - x_j\|_P - R_{ij})}{(\|x_i - x_j\|_P - r_{ij})^3}(x_i - x_j)^T P_{ij} & \text{if } R_{ij} > \|x_i - x_j\|_P > r_{ij} \\
\text{not defined} & \text{if } \|x_i - x_j\|_P = r_{ij} \\
0 & \text{if } \|x_i - x_j\|_P < r_{ij}
\end{cases}
\]  

(2.7)

By slightly modifying the notation of (2.7), the gradient of the avoidance function between an agent and a static obstacle takes the same form. Considering the overall collision avoidance objective for agent \( i \) as a sum of avoidance functions between cooperating agents and static obstacles, the gradient takes on the following form

\[
\frac{\partial v^a_i}{\partial x_i} = \sum_{j=1}^{N} \frac{\partial v^a_{ij}}{\partial x_i} + \sum_{k=1}^{N_k} \frac{\partial v^a_{ik}}{\partial x_i}, \ j \neq i, \ j = 1, \ldots, N, \ k = N + 1, \ldots, N + N_k.
\]  

(2.8)
2.3 Proximity and Flocking

Another objective relevant to autonomous multi-agent systems is that of the agents keeping proximity with respect to each other. This may be for the purposes of enforcing formation flight constraints or encouraging flocking or swarming behavior in the performance of some other objective. Proximity based objectives may also be used for modeling other distance-based constraints such as ensuring robust communications between agents over a specified range. Most if not all forms of communication rely on systems whose performance changes with respect to distance or line-of-sight. In this work, proximity control was implemented to represent a natural method for enforcing constraints associated with inter-agent wireless communications. Two different classes of proximity objective functions were used to model constraints resulting from the range of inter-agent communication systems.

2.3.1 One-Parameter Proximity

In the first proximity objective function, the ability of agent $i$ to communicate with agent $j$ is lost when the distance with respect to each other is greater than $\hat{R}_{ij}$. As in the formulation of the collision avoidance objective function (2.2), these communication system ranges are defined pairwise allowing for the modeling of non-homogenous communication capabilities in multi-agent systems. The communication capabilities of the agents were assumed to be uniform in every direction:

$$v_{ij}^p(x_i, x_j) = \max \left\{0, \|x_i - x_j\|^2 - \hat{R}_{ij}^2\right\}^2, \ i \neq j \quad (2.9)$$

where, $x_i$ and $x_j$ represent the position state variables of agent $i$ and $j$, respectively. In using this objective function, the control designer aims to decrease its value. A penalty is incurred for distances between agent $i$ and $j$ greater than $R_{ij}$. The gradient of the one-parameter proximity objective function (2.9) is straightforward to compute:

$$\frac{\partial v_{ij}^p}{\partial x_i} = 4 \max \left\{0, \|x_i - x_j\|^2 - \hat{R}_{ij}^2\right\} (x_i - x_j)^T. \quad (2.10)$$
Notably, the objective function in (2.9) penalizes distances greater than \( \hat{R}_{ij} \) between agents \( i \) and \( j \) yet it does not explicitly enforce a maximum distance between them. The one-parameter proximity function may be preferred in situations where constraining agents to remain within each other’s communication radius is not imperative to the success of other objectives.

2.3.2 Two-Parameter Proximity

An alternative nonnegative definite proximity function formulation which explicitly enforces a maximum inter-agent distance is defined as follows:

\[
v_{ij}^p(x_i, x_j) := \left( \max \left\{ 0, \frac{\hat{r}_{ij}^2 - \|x_i - x_j\|^2}{\|x_i - x_j\|^2 - \hat{R}_{ij}^2} \right\} \right)^2. \tag{2.11}
\]

Here, \( \hat{R}_{ij} \) and \( \hat{r}_{ij} \) represent the degradation and loss radii, respectively, between agent \( i \) and \( j \) where \( \hat{R}_{ij} > \hat{r}_{ij} > 0 \). With these two parameters, distances greater than the degradation radius are penalized as the control designer again aims to decrease (2.11). Furthermore, distances greater than \( \hat{r}_{ij} \) between agents \( i \) and \( j \) are explicitly prohibited. The control designer can choose these parameters after considering the range and performance characteristics of agent \( i \)'s communications system. In addition to the analogous form of the objective function in (2.2), \( \hat{P}_i \) like \( P_{ij} \) is a positive definite matrix which can be used to elongate the degradation and loss regions of agent \( i \).

The gradient of the two-parameter proximity function between agents \( i \) and \( j \) is given by

\[
\frac{\partial v_{ij}^p}{\partial x_i} = \begin{cases} 
0 & \text{if } \|x_i - x_j\|_{\hat{P}_i} \leq \hat{r}_{ij} \\
\frac{(\hat{R}_{ij}^2 - \hat{r}_{ij}^2)(\hat{r}_{ij}^2 - \|x_i - x_j\|^2_{\hat{P}_i})}{(|\|x_i - x_j\|^2_{\hat{P}_i} - \hat{R}_{ij}^2)^3} (x_i^p - x_j^p)^T \hat{P}_i & \text{if } \hat{r}_{ij} < \|x_i - x_j\|_{\hat{P}_i} < \hat{R}_{ij} \\
\text{not defined} & \text{if } \|x_i - x_j\|_{\hat{P}_i} = \hat{R}_{ij} \\
0 & \text{if } \|x_i - x_j\|_{\hat{P}_i} > \hat{R}_{ij}
\end{cases} \tag{2.12}
\]

Control effort calculations are unaffected by both the one- and two-parameter proximity objective functions when agent distances are less than \( \hat{R}_{ij} \) and \( \hat{r}_{ij} \), respectively.
For a multi-agent system with a simple topology in which all agents desire to remain proximal to all other agents, the proximity objective function for agent $i$ becomes a sum of its respective proximity functions

$$v^p_i = \sum_{j=1}^{N} v^p_{ij}(x_i, x_j), \ j \neq i. \quad (2.13)$$

Again, $N$ represents the number of agents of the multi-agent system. Correspondingly, the gradient of the overall proximity function is

$$\frac{\partial v^p_i}{\partial x_i} = \sum_{j=1}^{N} \frac{\partial v^p_{ij}}{\partial x_i}, \ j \neq i. \quad (2.14)$$

As was the case for the collision avoidance objective function (2.2), both the one- and two-parameter proximity functions are suitable for Lyapunov-like analysis as they are nonnegative definite and differentiable. Importantly, the topology of the proximity constraint on the multi-agent system will have implications when conducting such an analysis.

### 2.4 Dynamic Coverage

In contrast to the previous objectives considered thus far, dynamic coverage may be considered as a primary objective of multi-agent systems, along the lines of trajectory tracking or waypoint following. Coverage consists of the task of sensing or effecting a compact domain to a specified satisfactory level. Tasks that may be described by the following dynamic coverage formulation include search and rescue, fire suppression, mapping or surveillance, environmental measurement and analysis, agricultural service, minesweeping, and oil or other chemical spill cleanup. To accomplish these tasks, sensing may be accomplished by any of a variety of sensors, possibly denoted by electro-optical cameras or a broad class of sensors that measure different types of radiation. Effectors may include water or chemical dispersion mechanisms used to extinguish fires, mower blades and other agricultural tools, or effectors used to mitigate the dispersion of unwanted chemicals such as oil booms and skimmers. Sensing or effecting capabilities of the agents were modeled
with respect to the coverage domain as follows:

\[ S_i(p) = \frac{M_i}{R_i^2} \max \{0, R_i^2 - p\}^2, \quad p = \|x_i(t) - \bar{x}\|^2, \quad (2.15) \]

\[ \frac{\partial S_i(p)}{\partial x_i} = \frac{4M_i}{R_i^4} \max \{0, R_i^2 - \|x_i(t) - \bar{x}\|^2\} (x_i(t) - \bar{x})^T. \quad (2.16) \]

Here, \( M_i \) represents the peak sensing capacity and \( R_i > 0 \) the sensing radius of agent \( i \). This circular, normally distributed sensor model is depicted in Figure 2.1. For a system of \( N \) agents, the cumulative sensing function is then given by:

\[ Q(t, \tilde{p}) = \int_0^t \left( \sum_{i=1}^N S_i(\|x_i(\tau) - \tilde{p}\|^2) \right) d\tau, \quad (2.17) \]

in which \( \tilde{p} = [\tilde{x}, \tilde{y}] \in \mathbb{R}^2 \) represents a point within the coverage domain and \( x_i(\cdot) \) is agent \( i \)'s position in time. The dynamic coverage objective is modeled using an area integral of the time-dependent coverage error to represent the level at which the domain of interest is satisfactorily covered:

\[ e(t) = \int \int_D h(C^* - Q(t, \tilde{p})\phi(\tilde{p}))d\tilde{x}d\tilde{y}, \quad (2.18) \]
where \( h(z) \triangleq (\max\{0, z\})^3 \), \( D \) is the compact domain to be covered, \( C^* \) is a positive scalar representing satisfactory coverage, and \( \phi(\tilde{p}) \) is a nonnegative scalar function that can be used to include position dependent preferences or previous coverage information over the coverage domain. The coverage domain’s size and location according to a global reference frame are assumed to be known a priori. Within (2.18), the term (2.17) consists of a system of multiple agents whose sensing regions may overlap. In light of this fact and because the area integral of (2.18) is calculated over a time-dependent boundary and includes jumps at the critical intersection points of the robots’ sensing regions [40], an explicit form for the time derivative of the integral can be difficult to compute. This is worth considering for analyses that include gradient based controllers or Lyapunov stability analysis. When the agents’ sensing regions do not overlap, the following modified coverage error suffices:

\[
\hat{e}(t) = \int \int_D \frac{d}{dt}(h(C^* - Q(t, \tilde{p}))) \phi(\tilde{p})d\tilde{x}d\tilde{y}
\]

\[
= - \int \int_D h'(C^* - Q(t, \tilde{p})) \left( \sum_{i=1}^N S_i(\|x_i(t) - \tilde{p}\|^2) \right) \phi(\tilde{p})d\tilde{x}d\tilde{y}. \tag{2.19}
\]

Through simulation and experimentation, it has been noted that even when agent sensing regions overlap, the effect of the neglected integral terms are insubstantial.

The goal of dynamic coverage formulation is to drive the error term of (2.18) to zero, yet control effort does not appear in the modified derivative term of (2.19). Since this is a case of singular control, a modified second derivative of (2.18) is considered, again modified by neglecting terms related to the boundary and jumps:

\[
\hat{\dot{e}}(t) = \int \int_D h''(C^* - Q(t, \tilde{p})) \left( \sum_{i=1}^N S_i(\|x_i(t) - \tilde{p}\|^2) \right)^2 \phi(\tilde{p})d\tilde{x}d\tilde{y}
\]

\[
- 2 \sum_{i=1}^N \int \int_D h'(C^* - Q(t, \tilde{p})) S'_i(\|x_i(t) - \tilde{p}\|^2) \phi(\tilde{p})(x_i(t) - \tilde{p})^T \dot{x}_i d\tilde{x}d\tilde{y} \tag{2.20}
\]

Before conducting simulations and experiments, it was assumed that the agents would not leave the coverage domain while working to bring the coverage error to zero. In reality, the agents can drive the coverage error to
zero by moving their sensors outside of the coverage domain. To prevent this scenario from driving the error to zero, a modified version of the coverage error derivative in (2.19) was used:

$$\tilde{e}(t) = - \int_{D} (h'(C^* - Q(t, \tilde{p})) \left( \sum_{i=1}^{N} S^* - S_i(x_i(t) - \tilde{p})^2 \right) \phi(\tilde{p})d\tilde{x}d\tilde{y}),$$

(2.21)

where $S^* > \sum_{i=1}^{N} M_i$. The process used to define $\tilde{e}(t)$ follows the process used for (2.20):

$$\tilde{e}(t) = \int_{D} h''(C^* - Q(t, \tilde{p})) \left( \sum_{i=1}^{N} S_i(p) \right) \left( \sum_{i=1}^{N} S^* - S_i(p) \right) \phi(\tilde{p})d\tilde{x}d\tilde{y}$$
$$+ 2 \sum_{i=1}^{N} \int_{D} h'(C^* - Q(t, \tilde{p})) S'_i(p) \phi(\tilde{p})(x_i(t) - \tilde{p})^T \dot{x}_i d\tilde{x}d\tilde{y}.$$  \hspace{1cm} (2.22)

A key difference is that the sign of the second term in (2.20) and (2.22) differs. With $\dot{x}_i = [\dot{x}_{i1}, \dot{x}_{i2}]^T$, if the terms of (2.22) are labeled as $\tilde{e}(t) = a_0(t) + \sum_{i=1}^{N} a_{i1}(t) \dot{x}_{i1}(t) + a_{i2}(t) \dot{x}_{i2}(t)$, then the labels are:

$$a_0(t) = \int_{D} h''(C^* - Q(t, \tilde{p})) \left( \sum_{i=1}^{N} S_i(p) \right) \left( \sum_{i=1}^{N} S^* - S_i(p) \right) \phi(\tilde{p})d\tilde{x}d\tilde{y}$$

$$a_{i1}(t) = 2 \sum_{i=1}^{N} \int_{D} h'(C^* - Q(t, \tilde{p})) S'_i(p) \phi(\tilde{p})(x_{i1}(t) - \tilde{x})^T \dot{x}_{i1} d\tilde{x}d\tilde{y}$$

$$a_{i2}(t) = 2 \sum_{i=1}^{N} \int_{D} h'(C^* - Q(t, \tilde{p})) S'_i(p) \phi(\tilde{p})(x_{i2}(t) - \tilde{y})^T \dot{x}_{i2} d\tilde{x}d\tilde{y}.$$  \hspace{1cm} (2.23)

### 2.4.1 Static Obstacles in the Coverage Domain

In multi-agent systems with scenarios where collision avoidance with respect to static obstacles is an objective in addition to dynamic coverage, how to incorporate the obstacles into the coverage domain $D$ needs to be considered. While a technical detail, it is worth mentioning that in practice the coverage domain is discretized and often stored in an array. With this in mind, points within a priori known static obstacles’ avoidance regions were considered satisfactorily covered at the beginning of the dynamic coverage
objective. In scenarios where some static obstacles positions and or sizes are not known \textit{a priori} and instead exist in locations of the coverage domain, it may be reasonable to assume that the agents possess a sensing capability to distinguish between cooperative and noncooperative obstacles and therefore a capability to update the set of covered points accordingly.

2.5 Dynamic Global Coverage Coordination

The dynamic coverage objective function, based on the error term given in (2.18) and driven by a gradient-based controller, is susceptible to situations in which the error term becomes zero and yet not all points within the coverage domain are satisfactorily covered. In order to guarantee that all points will be satisfactorily covered, an additional objective is to be used together with the dynamic coverage objective. The proposed objective contrasts with other approaches for dealing with the limitations of the dynamic coverage objective, which bypass adding a constraint or an objective and instead use a conditionally switched controller \cite{42}. The proposed global coverage objective is designed so that if agent $i$’s nearest uncovered point $\tilde{p}_i^*$ is within its sensing region, the global coverage objective’s does not interfere with the dynamic coverage objective and is in fact inactive. This is motivated by the idea that if the change in cumulative sensing for the $i$th agent, $S_i(p)$, is zero and (2.18) is nonzero, then the agent should be directed towards the nearest uncovered point. A point is deemed uncovered if its level of coverage is less than $C^*$. The global coverage objective and dynamic coverage objectives are active at mutually exclusive times, similar to a switching controller.

When determining an agent’s nearest uncovered point, two complications arise. First, while covering the coverage domain at some arbitrary time, before satisfactory coverage has been achieved for all points, the set of uncovered points is dense and not necessarily convex. Determining the nearest uncovered point for agent $i$ can become part of an intractable search. This may be bypassed by discretizing the coverage domain, as is common in both simulation and experiments, to limit the search for the nearest uncovered point to one over a set of a finite size. Another complication that persists even after discretization of the set of uncovered points is that the nearest uncovered point may not be unique. This is mitigated by using the diamond-shaped
**Data:** Agent \(i\)'s position at time \(t\): \(x_i(t)\),
Coverage Domain \(D\) and its centroid \(\tilde{p}_c\),

Over approximation of largest distance between two points in \(D\): \(d_{max}\)

**Result:** The nearest uncovered point of agent \(i\): \(\tilde{x}_i^*\)

\(x = x_1(t), y = x_2(t)\);
\(\tilde{p} = \tilde{p}_c\);

**Algorithm 1:** Nearest uncovered point algorithm
search algorithm given in Algorithm 1. The algorithm is decentralized in nature and searches within $D$ from the position of agent $i$, selecting the first uncovered point found while searching progressively further from the agent in a diamond-like pattern. Notably, if the nearest uncovered point is not within a distance of $d_{max}$ from agent $i$, then the centroid of the coverage domain $\tilde{p}_c$ is taken as the nearest uncovered point for agent $i$, encouraging the robot to return to the coverage domain.

The global coverage objective for agent $i$ takes on a similar form as that of the sensing function used for the dynamic coverage objective (2.15):

$$v^g_i(x_i, x^*_i) = \frac{1}{(d_{max})^2} \max \left\{ 0, \|x_i - \tilde{x}_i^*\|^2 - R^2_i \right\}^2.$$  \hspace{1cm} (2.24)

This can be thought of as the sensing inability of agent $i$’s nearest uncovered point. Mentioned previously, $d_{max}$ is a design parameter chosen to approximate the maximum distance between any two uncovered points in the coverage domain $D$. Here, $R_i$ is as defined in (2.15) and accordingly, (2.24) is only nonzero when agent $i$’s nearest uncovered point is outside of its sensing radius. As soon as agent $i$ has completely covered all points within its sensing radius, its global coverage objective becomes active and remains active until agent $i$’s nearest uncovered point is within its coverage radius. As with (2.15), the dynamic coverage objective is nonnegative definite and differentiable

$$\frac{\partial v^g_i}{\partial x_i} = \frac{4}{(d_{max})^2} \max \left\{ 0, \|x_i - \tilde{x}_i^*\|^2 - R^2_i \right\} (x_i - \tilde{x}_i^*)^T.$$  \hspace{1cm} (2.25)
Chapter 3

Multiattribute Copulas and Utility Functions

The purpose of this chapter is to review relevant details of multiattribute utility function theory with salient details presented towards the end goal of designing controllers for a multi-agent system composed of \( N \) agents. Multiattribute utility functions are constructed in the decision analysis literature as an integral step in accurately representing the utility for decisions involving multiple attributes and uncertainty [10]. A multiattribute utility function \( U(v_1, \ldots, v_n) \) maps \( n \) attribute values \( v_1, \ldots, v_n, n \in \mathbb{N} \), to a single utility value. In this scalarization, as with all utility functions, more utility is preferred to less. It is often notationally convenient to write \( U(v_1, \ldots, v_i, \ldots, v_n) \) as \( U(v_i, \bar{v}_i) \) where \( v_i \) is the value of attribute \( i \) and \( \bar{v}_i \) represents the complement values of the remaining \( n-1 \) attributes. Multiattribute utility functions with the following properties will be considered within the scope of this chapter: continuous, bounded, nondecreasing with respect to each attribute, and for each attribute \( v_i \), there is at least one value of the complement values such that the function is strictly increasing in \( v_i \). With the end goal of using multiattribute utility copulas for the construction and design of multi-objective controllers for multi-agent systems, attributes of the multiattribute utility copulas will be represented by the objectives of the agents. The assumptions made regarding the multiattribute utility functions are reasonable in light of the fact the class of objectives considered in Chapter 2 are nonnegative definite and differentiable.

3.1 Multiattribute Utility Functions

Several methodologies for the creation of multiattribute utility functions exist. Many of the methodologies have forms which incorporate single-attribute utility assessments yet also make strong assumptions regarding the indepen-
dence of the attributes. This limits the form of the multiattribute utility functions to those of a multilinear combination

\[
U(v_1, \ldots, v_n) = \sum_{i=1}^{n} k_i U_i(v_i) + \sum_{i=1}^{n} \sum_{j>i} k_{ij} U_i(v_i) U_j(v_j) (3.1)
\]

\[
+ \cdots + k_{12\ldots n} U_1(v_1) U_2(v_2) \ldots U_n(v_n) (3.2)
\]

or under stricter assumptions, the multiplicative form

\[
U(v_1, \ldots, v_n) = \frac{1}{k} \left( \prod_{i=1}^{n} [1 + kk_i U_i(v_i)] - 1 \right). (3.3)
\]

Accordingly, methodologies that relax the independence assumptions have been developed to allow for the construction of multiattribute utility copulas of more general forms. Perhaps the methodology allowing for creation of the most general class of multiattribute utility functions is that of multiattribute utility copulas. Multiattribute utility functions constructed through the use of copula structures allow for the creation of a broad class of attribute dominance utility functions [8]. This class of utility functions must satisfy a set of properties originally required for the construction of multivariate probability functions from univariate marginal distributions. Some of these properties can be relaxed through the use of utility copulas allowing for the creation of an expanded class of multiattribute utility functions.

### 3.2 Copulas

Sometimes referred to as Sklar copulas, copulas were previously used to construct multivariate probability functions and in the development of a version of the triangle inequality suitable for probability theory [46]. These copula structures can be used to create joint probability functions by combining univariate marginal probability functions. For more information on copula structures, see [8, 46, 9]. The assumptions required and resultant properties of multiattribute utility copulas will be demonstrated by reviewing the preliminaries of two-dimensional copulas, then generalizing these properties to suit \(n\)-dimensional copulas and finally expanding through relaxation of these properties for the case of more general \(n\)-dimensional utility copulas.
3.2.1 Two Dimensional Copulas

A two-dimensional Sklar copula, or simply a 2-copula, \( C \) has the following properties:

- The domain of \( C \) is \( I \times I \) where \( I = [0, 1] \) and \( 0 \leq C(x, y) \leq 1 \).

- For every \( x, y \in I \)

\[
C(x, 0) = C(0, y) = 0, \quad C(x, 1) = x \quad \text{and} \quad C(1, y) = y.
\]

- For every \( x_1, x_2, y_1, y_2 \in I \) such that \( x_1 \leq x_2 \) and \( y_1 \leq y_2 \),

\[
C(x_2, y_2) - C(x_2, y_1) - C(x_1, y_2) + C(x_1, y_1) \geq 0.
\]

Here, relation (3.4) is referred to as the grounding property, relation (3.6) as the 2-increasing property of copulas, and \( C \) is a continuous mapping from \( I^2 \) to \( I \). These properties lead to the following theorem [46]:

**Theorem 1.** Let \( C \) be a 2-copula. Then for every \((x, y)\) in the domain of \( C \),

\[
\max (x + y - 1, 0) \leq C(x, y) \leq \min (x, y) \quad (3.7)
\]

**Proof:** Let \((x, y)\) be an arbitrary point in the domain of \( C \). From the definition of a copula, \( C(x, y) \leq C(x, 1) \) and \( C(x, y) \leq C(1, y) \). More explicitly, \( C(x, y) \leq C(x, 1) = x \) and \( C(x, y) \leq C(1, y) = y \) and so \( C(x, y) \leq x \) and \( C(x, y) \leq y \). It follows directly from this that \( C(x, y) \leq \min (x, y) \). Furthermore, (3.6) implies that \( C(x, y) \geq x + y - 1 \) which when combined with the fact that \( C(x, y) \geq 0 \) yields \( C(x, y) \geq \max (x + y - 1, 0) \). More explicitly, if \( x_2, y_2 = 1, x_1 = x, \) and \( y_1 = y \), then (3.6) becomes \( 1 - y - x + C(x, y) \geq 0 \). \( \square \)

3.3 \( n \)-Dimensional Copulas

Before characterizing the properties of copulas in the \( n \)-dimensional case, recall the definition of an \( H \)-volume:
Definition 1. Let $X_1, \ldots, X_n$ be nonempty subsets of $I$, and let $H$ be a function with domain $X_1 \times \cdots \times X_n$. Let $S = [x_{11}, x_{12}] \times \cdots \times [x_{n1}, x_{n2}]$ be a rectangle whose vertices are all in the domain of $H$ and such that $x_{i1} \leq x_{i2}, \forall i = 1, \ldots, n$. Then, the $H$-volume of $S$ is given by

$$V_H(S) = \sum \text{sgn}(c) H(c)$$

where $c = (c_1, \ldots, c_n)$ is a vertex of $S$, each $c_j$ is equal to either $x_{i1}$ or $x_{i2}$, and

$$\text{sgn}(c) = \begin{cases} 1, & \text{if } c_j = x_{j1} \text{ for an even number of } j \text{'s,} \\ -1, & \text{if } c_j = x_{j1} \text{ for an odd number of } j \text{'s.} \end{cases}$$

The generalized $n$-dimensional Sklar copula is a function of $n$ variables with the following properties:

- The domain of $C$ is $I \times I \times \cdots \times I = I^n$ where $I = [0, 1]$ and $0 \leq C(x_1, \ldots, x_n) \leq 1$.

- For every $x_i \in I$

  $$C(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) = 0,$$  
  $$C(1, \ldots, 1, x_i, 1, \ldots, 1) = x_i, \quad i = 1, \ldots, n.$$  

- For every $x_{i1}, x_{i2} \in I, i = 1, \ldots, n$, such that $x_{i1} \leq x_{i2} \forall i = 1, \ldots, n$ and where $[x_1, x_2] = [x_{i1}, x_{i2}] \times \cdots \times [x_{n1}, x_{n2}]$

  $$V_H([x_1, x_2]) \geq 0.$$  

As assumed earlier, the $n$-dimensional copula $C$ is a continuous mapping from $I^n$ to $I$. $C$ is labeled grounded if it satisfies (3.10) and $n$-increasing if it satisfies (3.12). These properties lead to the following theorem [46]:

Theorem 2. Let $C$ be an $n$-dimensional copula. Then for every $x_i, i = 1, \ldots, n$ in the domain of $C$,

$$\max (x_1 + \cdots + x_n - n + 1, 0) \leq C(x_1, \ldots, x_n) \leq \min (x_1, \ldots, x_n).$$

24
Proof: Let \((x_1, \ldots, x_n)\) be an arbitrary point in the domain of \(C\). From the definition of a copula, \(C(x_1, \ldots, x_n) \leq C(1, \ldots, 1, x_i, 1, \ldots, 1)\), for all \(i, i = 1, \ldots, n\). More explicitly, \(C(x_1, \ldots, x_n) \leq C(1, \ldots, 1, x_i, 1, \ldots, 1) = x_i\) for all \(i, i = 1, \ldots, n\), implying that \(C(x_1, \ldots, x_n) \leq x_i\) for all \(i, i = 1, \ldots, n\). It follows directly from this that \(C(x_1, \ldots, x_n) \leq \min(x_1, \ldots, x_n)\). \(\Box\)

3.4 Multiattribute Utility Copulas

Multiattribute utility copulas were developed by Abbas [10] in the field of decision analysis to create a broad class of multivariate utility functions through the combination of marginal utility functions. These multiattribute utility copulas extend the available forms of multivariate functions created through the use of Sklar copulas. A multiattribute utility copula \(C_\lambda(v_1, \ldots, v_n)\) of \(n\) variables, \(n \in \mathbb{N}\), is defined to have the following properties:

- \(C_\lambda\) is a continuous mapping from the \(n\)-dimensional hypercube \([0, 1]^n\) to the interval \([0, 1]\) and is normalized such that
  \[
  C_\lambda(0, \ldots, 0) = 0 \quad \text{and} \quad C_\lambda(1, \ldots, 1) = 1. \quad (3.14)
  \]

- \(C_\lambda\) is nondecreasing with respect to each of its arguments \(v_i\).

- For each argument \(v_i\), there exists a set of reference values \(\lambda_{i,j}\), with \(i, j = 1, \ldots, n, j \neq i\) such that
  \[
  C_\lambda(\lambda_{i,1}, \ldots, \lambda_{i,i-1}, v_i, \lambda_{i,i+1}, \ldots, \lambda_{i,n}) = c_i v_i + d_i \quad (3.15)
  \]
  with \(\lambda_{i,j}\) between 0 and 1, and \(0 < c_i \leq 1, 0 \leq d_i < 1\). For the reference value property \(C_\lambda(\cdot)\) in (3.15), if the reference value is defined at the minimum value of the complement attributes then \(\lambda = 0\) and the copula is labeled a Class 0 utility copula with the following property:
  \[
  C_0(0, \ldots, 0, v_i, 0, \ldots, 0) = c_i v_i + d_i, \quad i = 1, \ldots, n. \quad (3.16)
  \]
  Similarly, if the reference value is defined at the maximum value of the complement attributes, then \(\lambda_{ij} = 1\) and the utility copula is labeled a Class 1
utility copula with the following property:

\[ C_1(1, \ldots, 1, v_i, 1, \ldots, 1) = c_i v_i + d_i, \quad i = 1, \ldots, n. \tag{3.17} \]

Every continuous Sklar copula is a utility copula. Moreover, the relation

\[ C_1(v_1, \ldots, v_n) = a C(l_1 + (1 - l_1)v_1, \ldots, l_n + (1 - l_n)v_n) + b, \tag{3.18} \]

where \(0 \leq l_i < 1, a = 1/(1 - C(l_1, \ldots, l_n))\), \(b = 1 - a\), converts Sklar copulas into more general Class 1 multiattribute utility copulas. The utility copula \(C_1\) is grounded only if \(l_i = 0 \; \forall i = 1, \ldots, n\). When any subset of the \(l_i\) parameters are equal to zero, \(a = 1\) and \(b = 0\).

### 3.4.1 Archimedean Copulas

One of the relevant and important functional forms used to create multiattribute utility copulas is the extended Archimedean functional form. This functional form allows for a large variety of copulas to be constructed in a straightforward manner constructing copulas that possess many commonly desired independence properties. In the interest of constructing a Class 1 multiattribute utility copula, we consider an Archimedean functional form constructed using a multiplicative generator [10, 46]:

\[ C_1(v_1, \ldots, v_n) = a \psi^{-1} \left[ \prod_{i=1}^{n} \psi(l_i + (1 - l_i)v_i) \right] + b, \tag{3.19} \]

where

\[ a = \frac{1}{1 - \psi^{-1}(\prod_{i=1}^{n} \psi(l_i))}, \quad b = 1 - a \tag{3.20} \]

and in which \(0 \leq l_i < 1\). The generator \(\psi(\cdot)\) is equivalently referred to as the copula’s generating function. Generating functions are assumed to be continuous and strictly increasing for all \(x \in [0, 1]\), \(\psi(0) = 0\) and \(\psi(1) = 1\) ensuring that \(\psi(x)\) is monotonic and that its inverse is well-defined. Although the Archimedean functional form allows for a variety of generating functions,
the primary generating function considered was:

\[ \psi(x) = \frac{1 - e^{-\delta x}}{1 - e^{-\delta}}, \quad \delta \neq 0 \quad (3.21) \]

where \( \psi^{-1}(x) = -\frac{1}{\delta} \ln \left(1 - x(1 - e^{-\delta})\right) \). By varying the parameter \( \delta \), different trade-off functions among the attributes can be achieved. Determining all parameters of the multiattribute utility function is achieved by selecting the \( \delta \) parameter in accordance with preferences as well as using indifference probability assessments.

### 3.4.2 Determining Parameters of the Multiattribute Utility Copula

In order to compute values for the \( a, b, \) and \( l_i \) parameters of the Archimedean utility copula, the following \( n \) equations, one for each attribute, are used

\[ a(1 - l_i) = 1 - U(v_{i0}^0, \bar{v}_{i0}^0), \quad i = 1, \ldots, n. \quad (3.22) \]

Used together, the equations of (3.20) and (3.22) and a restriction that \( \sum_i U(v_{i0}^0, \bar{v}_{i0}^0) \leq 1 \) uniquely define all parameters of the Archimedean utility copula. Values for each \( U(v_{i0}^0, \bar{v}_{i0}^0), i = 1, \ldots, n, \) represent the probabilities obtained using indifference probability assessments. In an indifference probability assessment, the decision maker considers the scenario in which they are presented with a decision between two deals. The decision maker must determine at which probability he or she is just indifferent between taking 1) a deal consisting of the attribute set, and corresponding utility, they have already achieved \( C(v_{i0}^0, \bar{v}_{i0}^0) \); or 2) a deal with a binary outcome that achieves \( C(v_{i0}^*, \bar{v}_{i0}^*) \) with probability \( U(v_{i0}^0, \bar{v}_{i0}^0) \) or \( C(v_{i0}^0, \bar{v}_{i0}^0) \) with probability \( 1 - U(v_{i0}^0, \bar{v}_{i0}^0) \). This process determines the tradeoffs between the attributes of a multiattribute utility copula when it is constructed using the Archimedean functional form.

In the context of multi-objective control, where the attributes are in fact objective functions, the indifference probability assessments represent an intuitive and systematic method for determining the tradeoffs between objectives.
In previous work [2, 3], control strategies were developed for multi-player pursuit-evasion games which guaranteed evasion or capture by using strategies based on monotonic approximations to the minimum and maximum functions. Because the strategies were developed for multi-player systems, the resultant controllers were designed to accomplish multiple objectives. In the formulation, the objectives were represented by distance metrics representing satisfactory evasion of pursuers or capture of evaders. By using the minimum and maximum approximations, an effective scalarization of the multi-objective problem was accomplished. Through the use of the comparison principle, differential inequalities, and Lyapunov-like analysis, controllers designed using approximations to the minimum and maximum functions permitted determination of sufficient conditions for accomplishing the objectives.

In contrast to the control strategies developed for pursuit-evasion games, the primary interest of the control designs considered herein are for control of multi-agent systems intending to accomplish a full set of objectives discussed in Chapter 2. Instead of using goal functions constructed with both maximum and minimum approximations, subsets of objectives are chosen herein using only approximations to the maximum function and based on the number of objectives used to create a multi-objective function. Accordingly, focus will be on decreasing multiple nonnegative definite objective functions by using over-approximations to the maximum function as a scalarization of the set of objective functions. These scalarizations will be encapsulated in goal functions through which gradient-like control laws will be designed. Explicit relationships to the maximum function will be presented for goal functions constructed first using \( p \)-norms and then with multiattribute utility copulas as over approximations of the maximum function. These relationships will then be used to formally establish sufficient conditions for accomplishing all of an agent’s objectives in a multi-agent system. Control design using both
types of over-approximations to the maximum function will be discussed and the types of system models suitable for the controllers will be presented.

4.1 Maximum Approximation With $p$-Norms

Over-approximations to the maximum function are presented in [2], in addition to approximations of the minimum function from above and below and an approximation of the maximum function from below. The approximation of interest in this discussion is the over-approximation of the maximum function which is of a form also known as a $p$-norm. Multi-objective functions based on $p$-norms are suited for designing the goal functions of agents intending to accomplish an arbitrary number of objectives. For goal functions based on $p$-norms, the nonnegative definite objective functions are the functional arguments. This class of approximation functions also possess some monotonicity properties which can lead to a tunable tradeoff between computation efficiency and the tightness of the approximation. Results for simulations and experiments of multi-agent systems using $p$-norm controllers are presented in Chapter 5.

The maximum function may be approximated from above by using the following equation:

$$
\rho(\delta, v) = \delta \sqrt[n]{\sum_{i=1}^{n} v_i},
$$

with $\delta \in [0, \infty)$, $v = [v_1, \ldots, v_n]^T \in \mathbb{R}_+^n = [0, \infty)^n$, and where $n$ is a positive integer representing the number for objectives for a given agent. The maximum function from above can be defined in terms of the parameter $\delta$ or the vector of arguments $v$ as follows

$$
\rho_v(\delta) = \rho(\delta, v) \text{ for given } v \in \mathbb{R}_+^n, \\
\rho_\delta(v) = \rho(\delta, v) \text{ for given } \delta \in [0, \infty),
$$

where $\rho_p(v)$ is the $p$-norm of $v$ denoted by $\|v\|_p$ when $p = \delta \in [1, \infty]$. Let $M$ be defined as the index of the maximum argument of the vector of arguments $v$ or more specifically, $v_M = \max_{i \in \mathbb{N}} \{v_i\}$. This does not preclude situations
where the vector of arguments has a non-unique maximum argument as \( M \) can be chosen, without loss of generality, to be the index corresponding to any one of the maximum arguments. The over approximation of the maximum function in (4.1) is formally established and shown to satisfy a monotonicity property in the following theorem:

**Theorem 3.** The over-approximation to the maximum function (4.1) satisfies the following properties for an arbitrary positive integer \( n \) number of arguments:

\[
v_M \leq \rho_v(\delta_2) \leq \rho_v(\delta_1), \quad \forall \delta_1, \delta_2: 0 < \delta_1 \leq \delta_2 < \infty, \quad (4.3)
\]

\[
\lim_{\delta \to +\infty} \rho_v(\delta) = v_M. \quad (4.4)
\]

**Proof:** By first noting that the arguments of the over-approximation function (4.1) can be rewritten as

\[
\rho(\delta, v) = v_M \sqrt{1 + \sum_{i \neq M} \left( \frac{v_i}{v_M} \right)^\delta}, \quad (4.5)
\]

and using the following relationship

\[
\lim_{\delta \to +\infty} \sqrt{1 + \sum_{i=1}^{n} c_i^\delta} = 1 \quad \text{if } \forall i = 1, \ldots, n, \quad c_i \in [0, 1] \quad (4.6)
\]

with the fact that \( v_i/v_M \leq 1 \) for all \( i \in [0, \ldots, n] \), and \( n \geq 1 + \sum_{i \neq M} (v_i/v_M)^\delta \), it can be concluded that the equation (4.3) and (4.4) both hold.

Accordingly, the tightness of the approximation can be tuned by a choice of \( \delta \), coming at the expense of increased computation complexity generally associated with a larger \( \delta \) and corresponding to a tighter approximation.

The notation in (4.4) will be abused to stress the fact that goal functions are defined for each agent in the multi-agent system by using the notation \( \rho_i(v) \). This will also allow for the generalization of goal functions created using alternative forms.
4.1.1 Accomplishing All Objectives With $p$-norms

The goal functions considered in [6] cover scenarios in which agents are interested in accomplishing possibly different subsets of objectives. By making use of both types of approximations of the minimum and maximum functions, goal functions were designed to accomplish all objectives, a subset of the objectives, as well as objectives which were modeled with functions the agents wished to increase or decrease. The scope herein will be limited to scenarios in which the agents desire to decrease all of their respective objective functions. For all of the objective function formulations of Chapter 2 accomplishing a given objective amounts to decreasing the corresponding objective function’s value below some nonnegative scalar. Stated more precisely, agent $i$’s $j$th objective is accomplished if $v_{ij} \leq \varepsilon_{ij}$, $\varepsilon_{ij} \geq 0$. Accordingly, for agent $i$ to accomplish all $n_i$ objectives, its goal function $\rho_\delta(\cdot)$ must satisfy the following sufficiency condition:

$$\rho_\delta(v_{i1}, \ldots, v_{in_i}) \leq \min\{\varepsilon_{i1}, \ldots, \varepsilon_{in_i}\}. \quad (4.7)$$

In the case in which all $\varepsilon_{ij}$ are positive, a less stringent sufficiency condition is given by

$$\rho_\delta(\gamma_{i1}v_{i1}, \ldots, \gamma_{in_i}v_{in_i}) \leq 1, \text{ where } \gamma_{ij} = 1/\varepsilon_{ij}, j \in \{1, \ldots, n_i\}. \quad (4.8)$$

Clearly, when all $\varepsilon_{ij}$ are positive, the inequality (4.8) implies that each $v_{ij} \leq 1/\gamma_{ij} = \varepsilon_{ij}$ and ultimately that all objectives are satisfied. Alternatively, the goal function $\rho_\delta(\cdot)$ has accomplished all objectives if $\rho_\delta(\cdot) \leq \varepsilon$ where $\varepsilon := \min (\varepsilon_{i1}, \ldots, \varepsilon_{in_i})$. For information on more sophisticated scenarios in which agents wish to accomplish a subset of all objectives, see [6].

4.2 Maximum approximation With Multiattribute Utility Copulas

To develop the relationship between multiattribute utility copulas and over- approximations of the maximum function, several steps need to be taken. The bound established between general Sklar copulas and the minimum function given in Theorem 2 will be used to show a relationship between (3.19)
and the maximum function. In the case of multiattribute utility copulas, the function arguments are labeled attributes and when using these copulas as goal functions, the attributes will be represented by the nonnegative definite objective functions. Results for simulations and experiments of multi-agent systems using multiattribute utility copula controllers are also presented in Chapter 5.

The properties of multiattribute utility copulas given in Chapter 3 prescribe that the arguments must be bounded on the interval \([0, 1]\). In order to use general nonnegative definite objective functions like those given in equations (2.6), (2.13), (2.18), and (2.24), the following transformation is used:

\[
    w_{ij} = e^{-v_{ij}}, \quad v_{ij} \in [0, \infty), \quad w_{ij} \in [0, 1], \quad \forall i, j, \ i = 1, \ldots, N, \ j = 1, \ldots, n_i
\]

where \(N\) represents the number of agents and \(n_i\) represents the number of objectives for agent \(i\). By substituting the normalized objective functions of (4.9) into a general Sklar copula, and noting the bound in (3.13), the following relationship can be established

\[
    \max (v_{i1}, \ldots, v_{in_i}) \leq -\ln (C(w_{i1}, \ldots, w_{in_i})). \quad (4.10)
\]

It may be helpful to consider that in the case of positive definite arguments, \(v_i \in (0, \infty), \ i \in 1 \ldots, n\) the following relationship holds:

\[
    \frac{1}{\max (\frac{1}{v_1}, \ldots, \frac{1}{v_{n_i}})} = \min (v_1, \ldots, v_{n_i}), \ n_i \geq 2, \ n_i \in \mathbb{N}. \quad (4.11)
\]

The bound given in Theorem 2 can be extended to relate the maximum function to a multiattribute utility copula by first recalling the relationship between a Sklar copula \(C\) and a multiattribute utility copula \(C_1\)

\[
    C_1(w_{i1}, \ldots, w_{in_i}) = a_i C(l_{i1} + (1 - l_{i1})e^{-v_{i1}}, \ldots, l_{in_i} + (1 - l_{in_i})e^{-v_{in_i}}) + b_i, \quad (4.12)
\]

with \(0 \leq l_{i1}, \ldots, l_{in_i} < 1, \ a_i = 1/(1 - C(0, \ldots, 0)), \ b_i = 1 - a_i\). Using direct substitution, the following upper bound between agent \(i\)'s \(j\)th objective
function and the multiattribute utility copula $C_1$ holds

$$v_{ij} \leq -\ln \left( \frac{C_1(w_{i1}, \ldots, w_{in}) - 1 + a_i(1 - l_{ij})}{a_i(1 - l_{ij})} \right). \tag{4.13}$$

4.2.1 Accomplishing All Objectives With Multiattribute Utility Copulas

By again using $\varepsilon_{ij}$ to denote the value below which agent $i$’s $j$th objective is accomplished, a modified sufficiency condition is required to claim that all objectives are accomplished. In contrast to goal functions constructed using $p$-norms, which defined the $\varepsilon_{ij} = v_{ij}$, goal functions constructed using multiattribute utility copulas use a modified choice of $\varepsilon_{ij}$. First, the goal function for agent $i$ is defined in terms of the multiattribute utility copula:

$$\rho_i(x) = -\ln (C_1(w_{i1}(x), \ldots, w_{in}(x))). \tag{4.14}$$

If $\varepsilon_{ij}$ is defined as follows:

$$\varepsilon_{ij} = -\ln \left( \frac{C_1(w_{i1}, \ldots, w_{in}) - 1 + a_i(1 - l_{ij})}{a_i(1 - l_{ij})} \right), \tag{4.15}$$

then in the case of (4.14), to guarantee that all $n_i$ objectives of agent $i$ are accomplished, the following choice of $\varepsilon$ suffices:

$$\varepsilon = \arg \min_j \{-\ln (1 - a_i(1 - l_{ij})(1 - e^{-\varepsilon_{ij}}))\}. \tag{4.16}$$

The values for $\varepsilon_{ij}$ are given, and when $v_{ij} \leq \varepsilon_{ij}$, the $j$th objective of agent $i$ is satisfied. Selection of the $U(v_i^0, \bar{v}_i^*)$, via indifference probability assessments discussed in section 3.4.2, determine $a_i$ and the $l_{ij}$ parameters, and therefore influence the definition of $\varepsilon$. In the case that any of the $\varepsilon_{ij} = 0$, the choice of $\varepsilon$ is simplified, as in the case with a goal function constructed using a $p$-norm (4.7), to $\varepsilon = 0$. 
4.3 Control Using Goal Functions

This section presents a formal definition of accomplishing goals using the scalarizations given in the previous section. It will be important to first consider the class of agent dynamic models for which the proposed methodology to accomplish a goal is suited. The goal definition will take into account that the agent’s states evolve in time and therefore so do the goal functions. Equipped with both the formal definition of accomplishing a goal as well as the agent dynamic models, differential inequalities can be used to prove that the proposed methodology accomplishes the goal. Gradient-like controllers based on the goal function formulations using $p$-norms and multiattribute utility copulas will be presented with comments about their design.

Control strategies based on approximations to the maximum function avoid the difficulties of computing optimal control laws and instead, via differential inequalities, leverage an approach based on control Lyapunov functions [4, 5].

4.3.1 System Models

In general, it is assumed that the agents’ dynamic models are nonlinear yet affine in control

$$\dot{x}_i = g_i(x_i)u_i + h_i(x_i), \quad x_i(0) = x_{io}, \forall t \in [0, \infty), \quad i = 1, \ldots, N. \quad (4.17)$$

Differential equations like those in (4.17) are known to model a number of autonomous vehicles (denoted as agents) which is the main motivation as to their use in this formulation [47]. Here, $N$ represents the number of agents with $N = \{1, \ldots, N\}$. The states and control inputs of agent $i$ are denoted by $x_i \in \mathbb{R}^{n_i}$ and $u_i \in \mathbb{R}^{m_i}$, respectively and $x_{io}$ is the initial state of the $i$th agent. It is further assumed that within each agent’s set of states is a subset of states representing the agent’s position, all of which are controllable. Two specific instances were studied through the simulations and experiments of Chapter 5. In one instance, agent dynamics were modeled as linear single integrator systems, sometimes referred to as velocity systems

$$\dot{x}_i = u_i, \quad i = 1, \ldots, N \quad (4.18)$$
where \( x_i \) represents the position state variables of agent \( i \) and \( N \) is again the number of agents. In the other instance, agents were modeled using nonholonomic kinematic models

\[
\begin{align*}
\dot{x}_{i1} &= u_{i1} \cos (x_{i3}) \\
\dot{x}_{i2} &= u_{i1} \sin (x_{i3}) \\
x_{i3} &= u_{i2},
\end{align*}
\]

where again \( i = 1, \ldots, N \), \( N \) the number of agents and where \( x_{i1} \) and \( x_{i2} \) represent the \( i \)-th agent’s Cartesian coordinates and \( x_{i3} \) represents the agent’s angular position in (4.19). These models were particularly relevant to the simulations conducted in preparation for experiments on a system of multiple car-like robots.

Explicitly, accomplishing agent \( i \)’s goal is defined as follows:

**Definition 2.** A goal with corresponding goal function \( \rho_i(\cdot) \) is said to be accomplished at time \( T \) if a trajectory \( x_i(\cdot) \) of the system (4.17) satisfies \( \rho_i(T, x_i(T)) \leq \epsilon \) for a given value of \( \epsilon \) and for all \( t \in [0, T] \), with \( \|x_i(0) - x_j(0)\|^2_{P_{ij}} > r_{ij} \) for all \( j \), \( \rho_i(t, x_i(t)) < \infty \).

The definition above encompasses the most general of scenarios in which any of the objectives defined in Chapter 2 are to be accomplished.

The proposed way to guarantee that agent \( i \)’s goal is accomplished is to decrease \( \rho_i(\cdot) \) along the trajectories of (4.17) in time until \( \rho_i(T, x(T)) \leq \epsilon \) at some time \( t = T \). Because both goal functions defined using \( p \)-norms (4.1) and multiattribute utility copulas (4.14) are continuous and differentiable, the following minimization based control strategy for each agent \( i, i = 1, \ldots, N \), facing \( n_i \) objectives is proposed:

\[
\hat{u}_i(x) = \arg \min_{\|u_i\| \leq k_i} \left\{ \sum_{j=1}^{n_i} \partial \rho_i(x) \partial x_j h_j(x_j) + \sum_{j=1}^{n_i} \partial \rho_i(x) \partial x_j g_j(x_j) u_j \right\}
\]

\[
= \arg \min_{\|u_i\| \leq k_i} \left\{ \partial \rho_i(x) \partial x_i g_i(x_i) u_i \right\}
\]

\[
= -k_i g_i^T(x_i) \partial \rho_i(x) \partial x_i / \|\partial \rho_i(x) \partial x_i g_i(x_i)\|, \quad i = 1, \ldots, n_i,
\]

where \( \|u_i(t)\| \leq k_i, \forall t \in [0, \infty) \) and \( k_i > 0 \) is a scalar value representing a constraint on the control input for agent \( i \). In the case that \( \|\partial \rho_i(x) \partial x_i g_i(x_i)\| = 0 \)
the control can be set such that \( \hat{u}_i(x) = \gamma_i \) where \( \gamma_i \) is any constant vector of appropriate dimension that satisfies \( \|\gamma_i\| \leq 1 \). By choosing an appropriate \( \gamma_i \), a piecewise continuous control effort \( \hat{u}_i(x) \) can be constructed. For more details on the existence of solutions when using admissible, piecewise continuous control laws see [6] and the references reported therein. Alternatively, situations in which \( \|\frac{\partial \rho_i(x)}{\partial x_i} g_i(x_i)\| = 0 \) may be avoided by using the control law below

\[
\hat{u}_i(x) = -k_i g_i^T(x_i) \frac{\partial \rho_i(x)}{\partial x_i}, \quad i = 1, \ldots, n_i. \tag{4.21}
\]

In order to show that the control strategies in (4.20) and (4.21) achieve a given goal, the use of differential inequalities [3] suffices. Using the criteria specified in Definition 2, an \( \Omega(\cdot, \cdot, \cdot) : [0, \infty) \times \mathbb{R}^n \times \mathbb{R} \to \mathbb{R} \) that satisfies the following inequality must be determined:

\[
\frac{\partial \rho_i(t, x)}{\partial t} + \frac{\partial \rho_i(t, x)}{\partial x} \frac{\partial x}{\partial t} \leq \Omega(t, x, v(t, x)) \tag{4.22}
\]

for any \((t, x) \in [t_0, +\infty) \times \mathbb{R}^n\). The choice of \( \Omega(\cdot, \cdot, \cdot) \) is often simplified by using \( \dot{\rho}_i(\cdot) \), effectively leading to an equality in (4.22). To formalize a sufficient condition for achieving an agent’s goals according to Definition 2, the following theorem is recalled [6]:

**Theorem 4.** If the maximal solution \( \bar{z}(t) \) [48, 49] of the following differential equation:

\[
\dot{z}(t) = \Omega(t, x(t), z(t)), \quad z(0) = z_0 \tag{4.23}
\]

with initial condition \( z_0 > \rho_i(0, x_0) \) satisfies inequality (4.22), \( \bar{z}(T) \leq \epsilon \) for some \( T, T > 0 \), and is bounded for all \( t \in [0, T] \) then the goal is satisfied according to Definition 2.

**Proof:** The maximal solution satisfies the property in (4.23) from which it follows that \( \rho_i(t, x(t)) \leq \bar{z}(t) \) for any \( t \). Because \( \bar{z}(T) \leq \epsilon \), the relationship \( \rho_i(T, x(T)) \leq \bar{z}(T) \leq \epsilon \) holds and it follows that the goal is achieved at \( t = T \).  \( \square \)
4.4 Particulars of Goal Function Controllers

Having established the analytical tools required for using control laws based on gradients of the goal functions, several particularities arise and need to be addressed. These particularities are a result of both the goal function gradient-like control laws as well as the fact that all objective functions in Chapter 2 are specified in terms of agent position states. In the case of the gradient-like control laws, the gradient of the dynamic coverage objective, defined in (2.21), does not depend on control effort. In this case of singular control [50], the control methodologies in (4.20) and (4.21) were modified in one of two different ways. In the first modification, the goal function consisted of the avoidance and proximity objectives and appended to the ̂u_i(x) was the following dynamic coverage control law:

\[ u_c^i(t) = k_c^i g_i(x_i)^T a_i(t), \quad i \in N \]  

(4.24)

where \( k_c^i > 0 \) and \( a_i(t) \) is defined in (2.23). This resulted in the overall control law of

\[ u_i(t) = ̂u_i(t) + u_c^i(t). \]  

(4.25)

This was implemented in the simulations and experiments of Chapter 5 in which the goal function was defined using \( p \)-norms. In the other modification, the goal function consisted of all objectives: collision avoidance, proximity, dynamic coverage, and global coverage. The control law in (4.20) and (4.21) were modified in the following way. Using the chain rule, the gradient of the goal function is

\[ \frac{\partial \rho_i(x)}{\partial x_i} = \sum_{i=1}^{n_i} \frac{\partial \rho_i(x)}{\partial v_i(x)} \frac{\partial v_i(x)}{\partial x_i}. \]  

(4.26)

The gradient was altered to account for singular control in the dynamic coverage objective in the following manner:

\[ \frac{\partial \rho_i(x)}{\partial x_i} = \sum_{i=1}^{n_i} \frac{\partial \rho_i(x)}{\partial v_i(x)} b_i \]  

(4.27)
where \( b_i = \partial v_i / \partial x_i \) for the collision avoidance, proximity, and global coverage objectives. For the dynamic coverage objective, \( b_i = -a_i(t) \) a result of the fact that the control law of (4.24) intends to increase the second derivative (2.22) and that the control (4.20) or (4.21) intends to decrease the goal function along the trajectories of (4.17). This modification was implemented in simulations and experiments conducted in Chapter 5 where the goal function was designed using a multiattribute utility copula.

A particularity related to the definition of the objective functions arises in the implementation of goal based controllers when agents are modeled as nonholonomic systems. More specifically, (4.19) represents an under-actuated system in which control of the position states depends on control of a non-position state. In simulations and experiments of nonholonomic agents using goal functions both created using \( p \)-norms and created using multiattribute utility copulas, \( u_{i1} \) was specified as in (4.21) for \( p \)-norms and (4.20) for copulas, while \( u_{i2} \) was specified for both as below:

\[
u_{i2}(t, x) = -k_i^\theta (x_{i3} - \phi_i^o), \quad i \in \{1, \ldots, N\} \tag{4.28}\]

with \( k_i^\theta > 0 \) being a gain chosen to reflect agent \( i \)'s maximal angular velocity and \( \phi_i^o \) is defined as:

\[
\phi_i^o = \frac{\pi}{2} - \arctan \left( \frac{\partial p_i(x)}{\partial x_{i1}} \right). \tag{4.29}\]

The control laws in (4.20) and (4.21) are most efficient when the proportional controller in (4.28) drives agent \( i \)'s heading angle to the desired heading, \( \phi_i^o \). An alternative to the proportional controller given in (4.28) was one designed more as a bang-bang style controller

\[
u_{i2}(t, x) = -k_i^\theta \text{sign}\{x_{i3} - \phi_i^o(t)\}, \quad i \in \{1, \ldots, N\}. \tag{4.30}\]

In simulations and experiments where the control of (4.25) was used, a similar summation approach was used to account for the modified goal function and to determine a modified desired heading angle \( \phi_i^o \):

\[
h_i^o = \phi_i^o + \tilde{\phi}_i^o, \quad \tilde{\phi}_i^o(t) = \frac{\pi}{2} - \arctan \left( \frac{a_{i1}(t)}{a_{i2}(t)} \right). \tag{4.31}\]
4.5 Discussion of Goal Functions

The two goal function formulations presented in this chapter have differing qualities which may warrant one’s use over the other in appropriate situations. For both scalarizations, the goal functions permit explicit gradient-like control laws when the objective functions themselves are continuous and differentiable. Both the $p$-norm and multiattribute utility copula based goal functions allow for varying their tightness as over-bounds of the maximum function. For goal functions based on $p$-norms, tradeoffs or preferences amongst the different objectives were incorporated by using a modified $p$-norm [40] which depended on objective functions which were scaled by $\alpha_{ij} > 0$ as parameters to determine the tradeoffs between the objectives. Goal functions constructed using $p$-norms treat the objectives equally, because of the function’s form, with only the $\alpha_{ij}$ parameters allowing for tradeoffs among objectives to be incorporated. In practice, choosing the $\alpha_{ij}$ parameters requires consideration of what specific ranges the objective functions will take. Furthermore, the control designer is at a disadvantage when, for example, any of the design parameters of the objective functions from Chapter 2 are changed or if there are changes in the size of the coverage domain.

In contrast, goal functions constructed using multiattribute utility copulas result in a controller more robust to changes in objective function parameters and with a more straightforward method to select function parameters. This is a consequence of the fact that multiattribute utility copulas require the arguments to be normalized between 0 and 1 and the transformations of (4.9) are used to ensure this. The normalization of the objective functions allows for changes in objective function parameters without the need to modify the copula parameters $a_i, l_{ij}$, and $b_i$. Furthermore, these copula parameters are chosen through the intuitive and systematic method of indifference probability assessments. The copula form, specifically the Archimedean functional form considered in (3.19), allows more flexibility in determining the tradeoffs amongst the objectives. In general, Class 1 multiattribute utility copulas allow for constructing a larger variety of goal function forms through the use of different generating functions $\psi(\cdot)$ as well as through different functional forms.
Chapter 5

Simulations and Experiments

This chapter will present details and results related to both a MATLAB simulator and experimental testbed created to explore the implementation of controllers, constructed using the goal functions considered in Chapter 4, on multi-agent systems. Initially, the simulator was developed for testing basic scalarizations of multiobjective problems. Future simulations aimed to implement goal functions based on the $p$-norm (4.1), explore their implementation on different instances of dynamic models conforming to (4.17), and eventually led to the development of the multi-robot testbed for experiments on robotic systems. In response to the challenges and data resulting from implementing $p$-norm based controllers, simulations and eventually experiments were conducted using alternative goal function based controllers. By using multiattribute utility copula controllers, more extensive simulations and experiments were conducted allowing for the study of additional and alternative objective functions.

Before discussing the results of the various simulations and experiments, information and details related to the capabilities of both the simulation codebase and experimental testbed will be presented. For the simulations and experiments presented, controller and objective parameters will be presented along with the number of agents comprising the multi-agent system. Different sets of objectives were given to the agents or the multi-agent system in the various simulations and experiments and accordingly, will be enumerated. Insights and difficulties will be discussed as they relate to implementation and performance of multi-agent systems.
5.1 MATLAB Multi-Agent Simulator

A set of scripts capable of simulating agent performance according to the control laws discussed in Chapter 4 and intending to accomplish the objectives of Chapter 2 was developed in MATLAB R2011a on the Mac OS X operating system. A benefit of the scripts being developed using the MATLAB scripting language is that the simulation files are also capable of being run using different versions of MATLAB and on PCs running Windows based operating systems. The simulator was designed using an imperative programming methodology in which the core functionalities were broken out into various functions. These functionalities include control effort calculation, goal function construction and substitution, gradient calculations using the MATLAB Symbolic Math Toolbox, initialization and declaration, and plotting and data analysis. Plotting capabilities were designed by considering the objectives and associated metrics. A representative set of plots produced by the simulator is presented in Figure 5.1. This figure includes, from left to right, top then bottom, plots corresponding to inter-agent and inter-obstacle distances, level of coverage, thresholded level of coverage (for use with the nearest uncovered point algorithm), normalized coverage error, and agent trajectories. Within the coverage and thresholded coverage plots, satisfactory coverage is denoted by the color red while no coverage is denoted in blue, with a heat map coverage scale located on the east side of the figures. Notably, in all simulations units are normalized and therefore dimensionless except for time which is given in seconds. Further information regarding the data present in the simulation plots will be provided in subsequent sections.

In regards to architecture, the simulation scripts were designed with extensibility in mind, capable of simulating multi-agent systems of a varying number of agents and with the ability to add or modify objective functions. In practice, however, even with efforts to optimize the code for efficiency and memory use, simulations involving more than three agents resulted in considerably longer runtimes. Objectives can be assigned on a per agent basis with all objective and goal function parameters initialized and shared globally through a main script file.
5.2 Multi-Robot Testbed

In addition to the simulator, an experimental testbed was developed to provide a multi-robot system in which the various controllers could be evaluated on real world systems. Software implementing the goal function based control laws was developed for and run on four-wheeled car-like robots equipped with various sensors and custom electronics. By making use of a commercial off the shelf indoor motion capture system, the multi-robot system had access to position and orientation information of all cooperating robots within the testbed. Additional software was developed to document experimental data and provide a real-time visualization of robot distances, coverage domain information, and other data relevant to the multi-robot system objectives.

Experiments were conducted in the Mechatronics Laboratory in the department of Industrial and Enterprise Systems Engineering at the University of Illinois at Urbana-Champaign. Within the lab space, the OptiTrack motion capture system from Natural Point was utilized to determine and broadcast position and orientation data for the multi-robot system. The testbed is pictured in Figure 5.2, with the square coverage domain outlined in masking tape containing four robots and two silver cake pans which were used as obstacles. Position and orientation information from the motion capture system was broadcast at a rate of 100Hz and unique reflective marker arrangements on top of the robots were used by the motion capture system to provide individualized robot attitude information. An arrangement of eighteen infrared cameras formed the motion capture volume in which any robot
Figure 5.2: Multi-robot testbed

Figure 5.3: Example shared information flow of multi-robot testbed

with predefined marker arrangements could be localized. The motion capture software and a snapshot of the camera placement used to create the motion capture volume are picture in Figure 5.4. This information was broadcast over a local network to all robots as well as a computer displaying a real-time visualization of robot positions in the coverage domain and its corresponding level of coverage. Figure 5.3 diagrams the flow of shared information for the multi-robot system.

Robots were equipped with various electronics and sensors including custom printed circuit boards integrating wireless modems, an OMAP-L138 dual core processor with the ARM core running embedded Ångström Linux and the DSP core running DSP BIOS, an additional TMS320F28335 processor used for motor and other low-level control, and encoders to measure robot position and orientation. The localization data produced by the encoders
and the motion capture system was fused within a Kalman filter to extend the range of localization information and increase position and orientation accuracy. The four-wheeled, differential drive robots are driven using two DC motors powered by a pair of lithium polymer batteries. An example of the robots, their associated electronics, and motion capture marker arrangement used in the multi-robot experiments is presented in Figure 5.5. Robots did all control and objective function calculations using their onboard processors. This included inter-robot and inter-obstacle distance calculations, dynamic coverage map updating and fusion, and the search algorithm used in the global coverage objective. The robots were equipped with approximately 2GB of flash storage to which robot position and coverage maps were saved for later analysis.

Coverage and distance information was then broadcast to the visualization
software depicted in Figure 5.6. This software was run on a separate computer for the purpose of displaying the simulated sensing of the multi-robot system. For the experiments considered herein, the robots were not equipped with a specific sensor or effector but instead, the sensing capability was simulated using the sensor model given in (2.15). The robots were displayed in the visualization software using unique colors. Also displayed therein are the avoidance, detection, and proximity radii of all robots, in black, as well as indicators of each robot’s respective nearest uncovered point, colored the same as the corresponding robot. Coverage information was displayed using a heat map to represent the level of coverage within the coverage domain where red was used to denote satisfactory coverage and blue to represent no coverage, in the same manner as the plots produced using the MATLAB simulator. Also included for reference was the time duration of the coverage mission in the bottom righthand corner. Data collection consisted of the dedicated visualization software, data logging scripts run onboard the robots processors, and videos taken of the motion capture volume and visualization software. For more technical details and information on the testbed, robots, and associated software see [51].
5.3 Initial Simulation Development

Initially, simulations were conducted in which agents were modeled as single integrator systems, as in (4.18). Each agent’s overall control law was designed as a sum of the control efforts obtained from each objective’s corresponding gradient. Individual objectives were scaled using a tunable gain resulting in the following overall control law for each agent $i$, $i = 1, \ldots, N$ where $N$ is the number of agents:

$$ u_i(x_i) = u^a_i(x_i) + u^p_i(x_i) + u^c_i(x_i) $$

$$ = -g_i(x_i)^T \left( k^a_i \frac{\partial v^a_i}{\partial x_i} + k^p_i \frac{\partial v^p_i}{\partial x_i} - k^c_i(a_i(t)) \right). $$

(5.1)

where $u^a_i(x), u^p_i(x),$ and $u^c_i(x)$ represent the control efforts for the avoidance (2.6), proximity (2.9), and dynamic coverage objectives (2.18), respectively and $a_i(t)$ is defined as in (2.23). The gains $k^a_i, k^p_i,$ and $k^c_i$ were assumed to be nonnegative. This simulation served as a benchmark by which future controllers, more complex agent dynamics, and different multi-agent scenarios could be compared.

As depicted in Figures 5.8-5.10, three agents were assigned the task of

---

Footnote: This section includes results previously presented in [39]
dynamically covering a 32x32 square coverage domain. Amongst the three agents, one was designated as the leader and also tasked with avoiding static obstacles, not other agents. The other agents were tasked with remaining proximal to all other agents and to avoid collisions with each other and two static obstacles. Agents, with initial positions of (12,7), (14,15), and (24,7), and their trajectories are pictured in Figure 5.7 along side obstacles located at (7,12) and (16,25). Trajectories of agents 2 and 3 are depicted in black and blue, respectively. Agent avoidance and detection regions were circular while avoidance regions with respect to the obstacles were obstacle, and this is reflected in that the scaling matrices (2.5) with respect to the obstacles were $P_{ik} = \text{diag}\{0.5,1\}$. Avoidance and detection regions were homogenous with $r_{ij} = 1$ and $R_{ij} = 2$. Also, the proximity distance was $\hat{R}_{ij} = 8$. The peak sensing capacity and radii of the agents were specified with agent 1, shown with a magenta colored trajectory in Figure 5.7, as a leader agent. Accordingly, the peak sensing capacity of agent 1 was diminished with respect to the other agents yet agent 1’s sensing radius spanned the entire coverage domain: $M_1 = 1/75$, $M_2 = M_3 = 4$ and $R_1 = 32\sqrt{2}$, $R_2 = R_3 = 2$. The desired level of coverage was $C^* = 40$ and a uniform distribution of prior knowledge was assumed, $\phi(\tilde{x}) = 1$ for all $\tilde{x}$. By having a leader agent capable of sensing the entire coverage domain, the need for a global coverage objective or controller was not needed as in [41, 42].

The associated avoidance, proximity and coverage gains were $k_1^a = 0.015$, $k_2^a = k_3^a = 0.0015$, $k_2^p = 0.0001$, $k_3^p = 0.0004$, $k_1^c = 0.09$, $k_3^c = 0.0032$, and $k_3^c = 0.002$. Figure 5.8 shows the final agent positions with respect to the coverage domain which was satisfactorily covered after 5348s. This is also reflected in the coverage error functional vs. time plot given in Figure 5.9. Figure 5.10 shows inter-agent distances as well as distance between agents and obstacles. The avoidance, detection and proximity regions are also shown. Notably, coverage maps were kept on a per agent basis with the coverage summation of (2.17) calculated depending on whether or not agents were within each other’s proximity radius. More specifically, agents were considered vertices in a graph [52] in which edges existed between an agent pair as long as they were within a distance of the proximity radius of each other. Information was also exchanged as long as a spanning tree existed between two vertices.
Figure 5.8: Level of coverage and final agent positions over coverage domain

Figure 5.9: Global coverage error with respect to time
5.4  \( p \)-norm Goal Function

With the purpose of exploring control laws based on the goal functions discussed in Chapter 4 and of modeling car-like dynamic models, additional simulations were conducted in addition to experiments on the multi-robot testbed. These simulations and experiments focused on modeling the agent dynamics using the equations in (4.19). The multi-objective control laws were designed using goal functions based on \( p \)-norms (4.1). To deal with the particulars associated with the dynamic coverage error functional being a situation involving singular control, the translational velocity control law \( u_{i1}(t, x) \) was defined as in (4.25). The angular velocity control law \( u_{i2}(t, x) \) were chosen as in equation (4.28).

5.4.1  \( p \)-norm Simulations \(^2\)

A representative example of simulations conducted with the multi-agent system modeled with nonholonomic agents and under the control of a gradient-like, \( p \)-norm based control law is depicted in Figure 5.11-5.14. The multi-agent system scenario was defined very similarly to that described in Sec-

\(^2\)This section includes results previously presented in [40]
Figure 5.11: Level of coverage and final agent positions over coverage domain

Figure 5.12: Agent trajectories with respect to the coverage domain
tion 5.3 with all agents and obstacles having the same initial conditions and avoidance and detection shapes. With regard to the objective functions, the agents were given the same individualized tasks as earlier except that the leader agent was now also responsible for remaining proximal to the other agents. Furthermore, agent 2 and 3 were given a larger detection radius, $R_2 = R_3 = 3$, a larger proximity radius $R_2 = R_3 = 10$, and a higher peak sensing capacity $M_2 = M_3 = 6$. The key differences lie in the controller used to control the nonholonomic multi-agent system. The goal function gains $k_i$, as given in (4.21), were unity for all agents. The agents’ tradeoff coefficients, defined as in Section 4.5, for collision avoidance and proximity were $\alpha_{1j}^a = 0.015$, $\alpha_{2j}^a = \alpha_{3j}^a = 0.0015$ and $\alpha_{1j}^p = 0.0001$, $\alpha_{2j}^p = 3.5 \times 10^{-5}$, $\alpha_{3j}^p = 10^{-5}$, respectively with the $p$-norm effectively being a $2$-norm with $\delta = 2$. The angular velocity gain $k_{\theta i}^p$ was unity for each agent with the desired heading angle defined as in (4.31). With agent dynamic coverage gains of $k_{1i}^c = 2.5 \times 10^{-4}$, $k_{2i}^c = 5 \times 10^{-6}$, $k_{3i}^c = 2.5 \times 10^{-5}$, satisfactory coverage of the entire coverage domain occurred after 7919s. The agent trajectories, color-coded as in the plots of Section 5.3, take on a noticeably different, car-like form as evidenced in the hairpin turns of Figure 5.12. Also worth noting is that the trajectory of agent 3 spanned a larger distance within the coverage domain. The coverage error functional of Figure 5.13 again goes to zero, albeit after a longer period of time, and agents 2 and 3 again stop moving after entering local minima shortly into the mission timeline in Figure 5.14.

5.4.2 $p$-norm Controllers for the Multi-Robot Testbed

With the results from simulating nonholonomic agents controlled using goal functions constructed using $p$-norms, software was developed to implement similar controllers on the multi-robot testbed. Snapshots of an early version of the visualization software depicting an example experiment are given in Figures 5.15-5.17. The figures are displayed in order with the initial state of the coverage domain given in Figure 5.15 and the final robot configuration given in 5.17. In these figures, the visualization software is divided into two panels. On the lefthand side, the coverage domain is presented in nearly the same fashion as described in Section 5.2. A key difference is that coverage domain points within the obstacles’ avoidance regions were not removed from
Figure 5.13: Global coverage error with respect to time

Figure 5.14: Pairwise agent and agent to obstacle distances
the coverage domain and the obstacles are simply displayed as black circles. The second panel of this version of the visualization software was used to present distance information for the robots and obstacles with the proximity, detection, and avoidance regions given as dashed lines. The updated visualization software removed this panel and instead plotted the various radii around the robots in the primary panel. In contrast to the previous simulations of Sections 5.3 and 5.4.1, the robots were modeled homogeneously with no robot being capable of sensing the entire coverage domain. Because of this, and since there was no global coverage objective, the agents ultimately entered local minima and left portions of the coverage domain unsatisfactorily covered. A variety of experiments were conducted including those in which the multi-robot system was given similar tasking to that given in the previous simulations, Sections 5.3 and 5.4.1, and also including a scenario in which the leader agent was no longer responsible for the dynamic coverage task but instead tele-operated. Initial development of the multi-robot testbed did not include data logging scripts for analyzing at a later date experimental data.

Figure 5.15: Initial robot positions and distances
Figure 5.16: Intermediate robot positions and distances

Figure 5.17: Final robot positions and distances
5.4.3 Comments on $p$-norm Controller Implementations

Multi-agent simulator and experimental testbed implementations of control laws based on goal functions constructed using $p$-norms motivated research into two areas. First and foremost, a considerable amount of time was spent tuning the gains in (4.25) in conjunction with the tradeoff parameters of Section 4.5. Note that the gains and tradeoff parameters used in control of the multi-agent systems with both the single integrator models (4.18) and nonholonomic models (4.19) have values on the order of $10^{-4}$, plus or minus a couple orders of magnitude. Perhaps evidenced by the considerable variation in the gains amongst the different agents and different objectives, these gains were very dependent on objective function parameters like detection, proximity, and sensing radii. Changes in either the tradeoff parameters or objective function parameters necessitated the re-tuning of the other in order to achieve desirable multi-agent behavior. This becomes particularly relevant when the number of agents in the multi-agent system gets large as the number of parameters and gains to tune grows considerably. Having an intuitive and systematic procedure for selecting tradeoff parameters that is robust to changes in the objective function parameters will expedite tuning of the controller and allow for a quicker and more intuitive inclusion of additional objectives.

An additional research area inspired by work with $p$-norm controllers was incorporating an ability for the multi-agent system to continue dynamically covering the coverage domain even after entering local coverage minima. In [39, 6], designating a leader agent allowed for the multi-agent system to be controlled in a decentralized-like manner while assuring that the entire coverage domain would be covered satisfactorily. In [41, 42], a switching controller was used whenever agents entered local coverage minima to ensure that satisfactory coverage was achieved throughout the coverage domain. An alternative, switching-like approach which incorporates global coverage as an additional objective incorporated into a multiattribute utility copula based controller will be discussed in the sections that follow.
5.5 Multiattribute Utility Copula Controller

The goal of using multiattribute utility copulas in the design of goal function based control laws was to streamline the selection of design parameters and to develop a system for intuitively selecting the design parameters according to the relative importance of the various objectives. By making use of the analysis in Section 4.2, the tools and therefore basis for using multiattribute utility copula based control laws for multi-objective control were established. Preliminary simulations sought to verify the theoretical results associated with using multiattribute utility copula based goal functions as well as to see the effects of varying the tradeoffs between objectives. In these simulations, simplified scenarios in which agents desired to arrive at specified waypoints (2.1) while avoiding collisions with each other and static obstacles (2.6) were considered. After establishing the baseline performance of multiattribute utility copula controllers, this type of controller was adapted for use in multi-agent scenarios similar to those considered in Sections 5.3, 5.4.1, and 5.4.2. Simulations were conducted using agents’ dynamics modeled as both single integrator 4.18 and nonholonomic systems 4.19 and included the incorporation of a global coverage objective. Finally, multiattribute utility copula based controllers incorporating the global coverage objective were implemented on the multi-robot testbed.

5.5.1 Preliminary Results

To study a baseline for the performance of a multiattribute utility based copula controller, a simple two agent scenario was considered. Pictured in Figures 5.18 and 5.19, two agents, modeled as in (4.18), were given symmetric initial positions, denoted by triangles at (5,17) and (25,17), relative to each other as well as two elliptical static obstacles positioned at (10,16) and (20,16). The agents’ respective waypoints are denoted by squares at (25,16) and (5,16). Avoidance and detection regions were \( r_{ij} = 1 \) and \( R_{ij} = 2 \), respectively with the scaling matrices of the obstacles defined as \( P_{ik} = \text{diag}\{0.5,1\} \). The multiattribute utility copula controller gains for the agents were \( k_i = 5.0 \) and the agents’ goals were considered accomplished when their goal function value was such that \( \rho_i(x) \leq \epsilon = 0.001 \). For the objective function tradeoff parameters, agent 1 favored waypoint following and agent 2 collision avoidance
with the agents’ $l_i$’s specified as follows: $l_{11} = 0.01$, $l_{12} = 0.99$, $l_{21} = 0.99$, $l_{22} = 0.01$.

Ultimately, this led to agent 1 and 2 arriving at their waypoints after 18.3s and 20.4s, respectively. Agent trajectories relative to each other and the static obstacles are shown in Figure 5.18 and inter-agent and agent-obstacle distances for the duration of the simulation are given in Figure 5.19. These figures clearly demonstrate the ability for the agents to satisfy both objectives simultaneously, evidenced by the waypoint distances approaching zero and avoidance distances remaining larger than the avoidance radius. Furthermore, the ability of the multiattribute utility copula controller to systematically and intuitively specify preferences amongst the objectives is demonstrated by a shorter goal achievement time of agent 1 with respect to agent 2.

5.5.2 Indifference Probability Assessments

The use of the multiattribute utility copula based controller was extended to a scenario involving three single integrator agents in Figures 5.20 and 5.21. In contrast to the baseline simulation, agents 1 through 3 were given arbitrarily specified initial conditions, (6, 27), (21, 10), and (22, 27), and

\footnote{This section includes results previously presented in [31]}
waypoints, (16, 13), (15, 30), and (14, 15), respectively. The same static obstacles remained elliptical as before but now positioned at (10, 20) and (17, 19). Agent controller gains were $k_1 = 0.5$ and $k_2 = k_3 = 1$. In contrast to the baseline simulation, tradeoff parameters were determined using the indifference probability assessments, defined in Section 3.4.2.

To elaborate on the indifference probability assessment process of the simulations in Figures 5.20 and 5.21, $U(w_{i1}^o, \bar{w}_{i1}^o)$ were chosen based on a decision between two scenarios. In one scenario, collisions would not occur yet agent $i$ would remain distant from its waypoint. Alternatively, the agent would arrive at its waypoint in a collision free trajectory with probability $U(w_{i1}^o, \bar{w}_{i1}^o)$ or collide and remain distant from its waypoint with the remaining probability. In comparison, $U(w_{i2}^o, \bar{w}_{i2}^o)$ was specified by choosing between two similar scenarios. The first scenario was one in which agent $i$ arrived at its waypoint but did not accomplish its collision avoidance objective. The second scenario involved agent $i$ arriving at its waypoint without collisions with probability $U(w_{i2}^o, \bar{w}_{i2}^o)$ or experiencing a collision and remaining distant from its waypoint with the remaining probability. The chosen indifference probabilities for the simulation in Figure 5.20 represent a control law design in which the waypoint following objective is preferred over the collision avoidance objective. The probabilities of Figure 5.21 depict results of agents strongly preferring collision avoidance over waypoint tracking.
Figure 5.20: $U(w_{11}^a, \bar{w}_{11}^s) = 0.95$, $U(w_{12}^a, \bar{w}_{12}^s) = 0.01$, $\epsilon = 0.001$
Figure 5.21: $U(w_{i1}, \bar{w}_{i1}) = 0.1, U(w_{i2}, \bar{w}_{i2}) = 0.7, \epsilon = 0.001$
Specifying the objective preferences based on the discussed assessments had varying affects on the amount of time it took for agents to accomplish their goals, inter-agent distances, and agent trajectories. In both simulations, agent 1 arrived at its waypoint fastest when collision avoidance was preferred over waypoint following yet agent 2 arrived at its waypoint fastest when waypoint following was preferred. In Figure 5.21, trajectories of agent 1 and 2 demonstrate a preference towards the collision avoidance objective as they approach their respective waypoints.

The corresponding utility values for two different simulations are given in the captions of Figures 5.20 and 5.21 along with the corresponding goal function threshold. Agents 1, 2, and 3 accomplished their goals in the simulation of Figure 5.20 with elapsed times of 39.5s, 14.4s, and 21.4s, respectively. The simulation of Figure 5.21 resulted in elapsed times of 23.1s, 14.3s, and 25.5s for the agents to accomplish their respective goals. In both figures, the trajectory of agent 1, its desired waypoint, its distances relative to the other agents and static obstacles, and its corresponding $\rho_i$ value are depicted in magenta. Agents 2 and 3 have the same data depicted in blue and cyan, respectively. Within the figures, from top to bottom and left to right, plots corresponding to agent trajectories, all inter-agent, agent-obstacle, and agent-waypoint distances, and goal function values versus time are pictured. In each figure’s distance plot, agent distance from its respective waypoint is denoted in red with a unique marker.

5.6 Global Coverage Objective

Simulations were conducted to demonstrate the effectiveness of the control laws designed using the multiattribute utility copula constructed in (3.19) for multi-agent systems intending to accomplish dynamic coverage. Both single integrator and then nonholonomic models were simulated with Figures 5.22, 5.24, 5.26, and 5.30 corresponding to single integrator models (4.18) and Figures 5.23, 5.25, 5.27, and 5.31 to nonholonomic models as in (4.19). In both simulations, a system of three robots were given the objective of dynamically covering a square domain with a side length of 10, units normalized, as well as the global coverage objective given in (2.24). Importantly, the two-dimensional coverage domain was discretized in both dimensions at an
The agents were again given the objective of avoiding collisions with one another as well as two elliptical obstacles, denoted with indices 4 and 5, respectively, located at (3,5) and (7,2). Avoidance and detection regions for inter-agent collision avoidance were assumed to be circular, while for collision avoidance between agents and static obstacles, the obstacles were shaped
Figure 5.23: Nonholonomic agent trajectories

as ellipses with scaling matrices $P_{i4} = P_{i5} = \text{diag}\{1,0.5\}$. Avoidance and detection radii were specified as though the agents were homogenous with $r_{ij} = 0.1$ and $R_{ij} = 0.6$. Distances for inter-agent and agent and obstacle pairs for the duration of the simulation are given in Figure 5.24 for the single integrator models and in Figure 5.25 for the nonholonomic models. Notably, in neither simulation do the agents come within a distance less than or equal to the avoidance radius with respect to each other or the static obstacles at any point during the simulation.

Furthermore, all agents were given the objective of remaining proximal to all other agents so as to model constraints for robust communications. The communications capabilities of the agents were assumed homogenous and given a proximity radius of $\hat{R}_{ij} = 6$ for the single integrator models and $\hat{R}_{ij} = 8$ for the nonholonomic models. Within these distances, agents were allowed to share their respective coverage maps. Communication capabilities were further assumed to allow for sharing of agent coverage maps as long as a spanning tree existed between communicating robots. The agents were limited to maximum translational, for both models, and rotational, for the nonholonomic models, velocities of $k_i = k_i^o = 1$. More specifically, the multiattribute utility copula controllers were chosen as given in (4.14) for the
Figure 5.24: Pairwise agent and agent-obstacle distances, single integrator models

Figure 5.25: Pairwise agent and agent-obstacle distances, nonholonomic models
goal function. Accordingly, lateral velocity input was defined as in (4.20) for both system models, with the goal function gradient modification of (4.27), and for the nonholonomic models, angular velocity input was defined as in (4.30) with $\phi_i^o$ as in (4.29).

To determine the parameters of the multiattribute utility copula, the $a_i, b_i, l_{ij}$ parameters, indifference probability assessment values of $U_{i1} = 0, U_{i2} = 0.7, U_{i3} = 0.1, \text{and } U_{i4} = 0.1$ for collision avoidance, proximity, dynamic coverage, and global coverage, respectively were used for each agent modeled as a single integrator system. For the nonholonomic systems, indifference probability assessment values of $U_{i1} = 0.3, U_{i2} = 0.65, U_{i3} = 0.01, \text{and } U_{i4} = 0.01$ for collision avoidance, proximity, dynamic coverage, and global coverage, respectively were used. A value of $\delta = 1$ was used for the generating functions (3.21) used to create the multiattribute utility copulas.

Coverage error according to the coverage map of agent 1 is given in Figure 5.26 and Figure 5.27 for the single integrator and nonholonomic modeled agents, respectively, to show how the coverage error decreased in time. Additionally, agent goal function values for the simulations in which the agents were modeled as single integrator systems and as nonholonomic systems are
shown in Figure 5.28 and Figure 5.29, respectively. At a time of approximately 214s the nonholonomic multi-agent system had accomplished their respective goals by satisfactorily covering the entire coverage domain, remaining collision free, and finishing in the relative locations shown in Figure 5.31. Similarly, for the multi-agent system of single integrator agents, the goal function was accomplished after about 157s.

5.7 Experiments With Copula-Based Goal Function

To further validate controllers constructed using multiattribute utility copulas and the global coverage objective given in (2.24), software was developed to conduct additional experiments with the multi-robot testbed. A series of experiments were conducted with various numbers of robots, obstacles, initial conditions, and controller gains. Two specific examples are presented herein with the trajectories of a four robot experiment shown in Figure 5.32 and a three robot experiment shown in Figure 5.33. For the four robot experiment, robots were initially positioned around the northeast corner while the three robot experiment started with robots placed just north of the northern
Figure 5.28: Single integrator agents’ goal functions versus time

Figure 5.29: Nonholonomic agents’ goal functions versus time
Figure 5.30: Final single integrator agent configuration

Figure 5.31: Final nonholonomic agent configuration
most obstacle. In general, collision avoidance followed the definition (2.6). Obstacles were circular to model the physical obstacles used in the testbed and in contrast to the previous simulations and experiments, avoidance radii for the static obstacles differed from the avoidance radii for the agents.

For the four robot and three robot experiments presented herein, the detection radii were $R_{ij} = 0.65\text{m}$ for inter-robot avoidance and $R_{ik} = 0.5\text{m}$ for robot-obstacle avoidance. The avoidance radii were $r_{ij} = 0.25\text{m}$ for both inter-robot and robot-obstacle avoidance. Various proximity objective functions were used, including the one defined in (2.11) as well as variations on the objective function definition of (2.9) in which the robots desired only to remain proximal to a subset of the multi-robot system. In the experiments presented herein, however, the proximity objective was described as in (2.9) with the robots desiring to stay proximal to all other robots. Distances between the robot pairs, also in relation to the detection, avoidance, and proximity radii, for the three and four robot experiments are pictured in Figures 5.35 and 5.34, respectively. Similarly, robot-obstacle distances for the two experiments are given in Figures 5.36 and 5.37. Again, the multiattribute utility copula controllers followed the definitions given in (4.14) for the goal function, with the robots again modeled as nonholonomic sys-
Accordingly, lateral velocity input was defined as in (4.20), with the goal function gradient modification of (4.27), and angular velocity input as in (4.30) and $\phi_i^\theta$ as in (4.29). The corresponding gains for the robots were $k_i = 0.7$ and $k_i^\theta = 1$. The copula was based on the Archimedean functional form given in (3.19) with a generating function (3.21) parameter of $\delta = 1$.

The testbed’s coverage domain, outlined in masking tape in Figure 5.2, was modified to be a square with an approximate length of 4.5 meters. The domain was discretized into a grid of approximately 160x160 points with each point having a dimension of $0.00079m^2$. A proximity radius of 3m was selected for both experiments presented herein and identical indifference probability assessments were used for both experiments. For the collision avoidance, proximity, dynamic coverage, and global coverage objectives, the corresponding utility values were $U_{i1} = 0.01$, $U_{i2} = 0.97$, $U_{i3} = 0.001$, and $U_{i4} = 0.01$. These parameters were selected to reflect the preference of dynamic coverage over everything else, with collision avoidance and global coverage also highly preferred over proximity. For the dynamic coverage objective, each robot used the search algorithm defined in Algorithm 1 with $d_{max} = 6.364m$ defined as the diagonal of the coverage domain. In both experiments, the circular obstacles, denoted in black, were placed in
Figure 5.34: Pairwise robot distances

Figure 5.35: Pairwise robot distances
roughly the same locations as noted in the trajectory plots of the four robot, Figure 5.32, and three robot, Figure 5.33, experiments. In the four robot experiment, the robots satisfactorily covered the entire coverage domain after approximately 97s. For the three robot experiment, the robots had satisfactorily covered the entire coverage domain at a time of 149s.

5.7.1 Comments on the Multiattribute Utility Copula Controller and Global Coverage Objective

Through the various simulations and experiments, the strengths and weaknesses of using gradient-like control laws based on goal functions constructed using multiattribute utility copulas became apparent. Copula based control laws mitigated the challenges associated with accurately determining the tradeoffs between objectives. By using the indifference probability assessment values given in Section 5.7, the influence of the proximity objective on the multi-agent system behavior was reduced in a straightforward manner. By requiring the probability value to be large, the other objectives were allowed to take precedence in situations where the objectives competed with each other. By following the design procedures for multiattribute utility cop-
ulas, the search for an appropriate tradeoff value was simplified by looking in the set \([0, 1]\). Furthermore, the intuitive and systematic procedure for determining the tradeoffs amongst the objectives allowed for a timely incorporation of the additional global coverage objective \((2.24)\) as well as experiments with additional robots. Consideration should be made when using objective functions, especially in discrete implementations, which may regularly reach large values. Numerical limits associated with using the transformations in \((4.9)\) can lead to situations in which the objective effectively becomes inactive in the multiattribute utility copula controller as \(w_{ij} \approx 0\). This can be mitigated by considering the range of the corresponding objective function \(v_{ij}\), possibly pre-scaling its values, and by considering the time step of the discrete control law implementation.

The global coverage objective allowed for distributed control of a multi-agent system without the need for a leader agent with superior dynamic coverage capabilities. It also allowed for situations in which it is impractical, by cost or sensing capability, to have a leader agent capable of sensing the entire coverage domain. Encoding the objective in this manner \((2.24)\) also represents an alternative to the switching control law used in \([41, 42]\). The search for the nearest uncovered point, given in Algorithm 1, represents a
basic but potentially inefficient method for ensuring the entire coverage domain is satisfactorily covered. It differs from the methodology for coordinated global coverage in [17] which coordinates a global coverage control using an elaborate and centralized scheme. What is lost in efficiency may, however, be regained in the ease of implementation. While the global coverage objective does not interfere with the dynamic coverage objective, depending on the proximity objective formulation used, it may be quite likely that the global coverage and proximity objectives conflict with each other.
Chapter 6

Conclusions

This dissertation has presented contributions in the area of multi-objective control and coordination of multi-agent systems. Constraints and behaviors related to collision avoidance, proximity, and dynamic coverage were treated through the use of both established and new nonnegative definite objective functions. The multiattribute utility copula, a tool used in the field of multiattribute decision analysis, was adapted for use in multi-objective control because of its ability to intuitively and systematically assign preferences amongst attributes. Examples of gradient-like control laws constructed with functions designed as over approximations of the maximum function were studied. Formal methods for verifying that all objectives constituting a goal function are accomplished were provided through the application of theory from differential inequalities and Lyapunov-like analysis. The relative ease with which such control laws can be implemented was demonstrated through their use in a variety of multi-agent simulations and experiments.

Chapter 2 provided building blocks with which common multi-agent behaviors can be modeled and through various combinations of these objectives, complex behaviors can be produced. The objective functions’ suitability for heterogenous systems was emphasized by noting that parameters are chosen on a per agent or agent pair basis. Collision avoidance was presented for the similar yet separate situations of cooperatively avoiding other agents and for avoidance of static obstacles. An explicit example for creating elliptical avoidance and detection regions was given. For constraints related to the communication capabilities of the agents, proximity objectives that incorporate soft or hard constraints were presented. Details of an effective coverage control scheme were presented and a novel approach for ensuring global coverage was discussed. The global objective was defined to be active only at times when the dynamic coverage objective is inactive, and followed the nonnegative definite formulations of the other objective functions. These
formulations allowed for explicit gradients amenable to the control methodologies discussed in Chapter 4.

Before constructing multi-objective goal functions, Chapter 3 provided the appropriate preliminaries and review of multiattribute utility copulas. Original uses and resulting properties of copula structures were presented. A detailed construction of an \( n \)-dimensional multiattribute utility copula was demonstrated after carefully considering less general copulas and 2-dimensional utility copulas. Explicit proofs establishing copulas as under approximations of the minimum function were provided towards the eventual use of copulas as over approximations of the maximum function. The multiattribute decision theory tool of indifference probability assessments was discussed in terms of its use to determine the tradeoffs between attributes and by extensions, agent objectives.

Chapter 4 explored the theoretical tools necessary for designing control laws using over approximations of the maximum function. A formal definition of a goal, which consists of multiple objectives, was presented. By defining goal functions based on over approximations to the maximum function, explicit conditions for satisfying all objectives could be given. Goal functions constructed using \( p \)-norms and multiattribute utility copulas were discussed and their use as over-approximations of the maximum function was formally developed. In the case of multiattribute utility copulas, a transformation required to use general nonnegative definite objective functions as attributes was presented. Finally, suitable classes of dynamic systems were considered and explicit control laws based on the goal functions were constructed. Details related to the gradients of the various objective functions were discussed and possible accommodations to the control laws were presented. A formal proof of how the proposed control laws can be used to accomplish the agent goals was presented.

The research presented herein culminated in applications of the considered objectives and control methodologies in the simulator and testbed discussed in Chapter 5. Implementations on a number of different multi-agent systems were presented ranging from scenarios with two agents facing two objectives to scenarios in which four robots faced four objectives. Heterogenous and homogenous multi-agent system implementations were considered in which the agents were modeled as single integrator systems or nonholonomic systems. The ability of goal function based, gradient-like control laws to ac-
complish multiple objectives was validated through the performance of the various simulations and experiments. More specifically, the developed multiattribute utility copula based control laws were implemented in a straightforward manner and the novel dynamic coverage objective was demonstrated in both simulation and experiments.

The provided simulations and experiments demonstrate, through the figures of Chapter 5, the feasibility of the objective functions presented herein to be used for multi-objective control and coordination of simple and complex multi-agent systems. Combined with the explicit form control laws, the framework presented in this dissertation represents a viable tool for researching a multitude of multi-agent problems.

6.1 Future Research

The research presented in this dissertation may encourage further research as the class of problems considered herein could be expanded in a number of areas. Increasing the scope of considered problems could consist of increasing the number of agents, defining additional and more specific objective functions, exploring additional multiattribute utility copula formulations and generating functions, exploring the performance of various systems relative to preferences and size, and developing theoretical results for guaranteeing accomplishment of goals consisting of various sets of objectives.

The suitability for the framework presented herein for multi-agent systems of different sizes or groups of multi-agent systems remains an avenue to explore. Underlying system and objective assumptions could be articulated further and one area where this is particularly relevant is in the proximity objectives considered herein. Additional formulations, especially as they relate to agent topologies and communication, may provide insight into different types of system behavior or theoretical guarantees for accomplishing goals. A strong interest exists in considering different sensor models and implementations for the dynamic coverage objective. The global coverage objective considered herein is one of many possible methodologies for encouraging global coverage and its performance relative to other methodologies could be explored further. Another interesting area for future research is situations where agents intend to accomplish different combinations of the objective
functions.

Finally, multiattribute utility copulas represent a broad class of multi-objective scalarizations. Additional functional forms suitable as goal functions for multi-objective control could be explored especially as they relate to independence among the attributes. Perhaps the largest area for future research includes conducting a sensitivity analysis of multiattribute utility copulas with respect to the performance of multi-agent systems.
References


