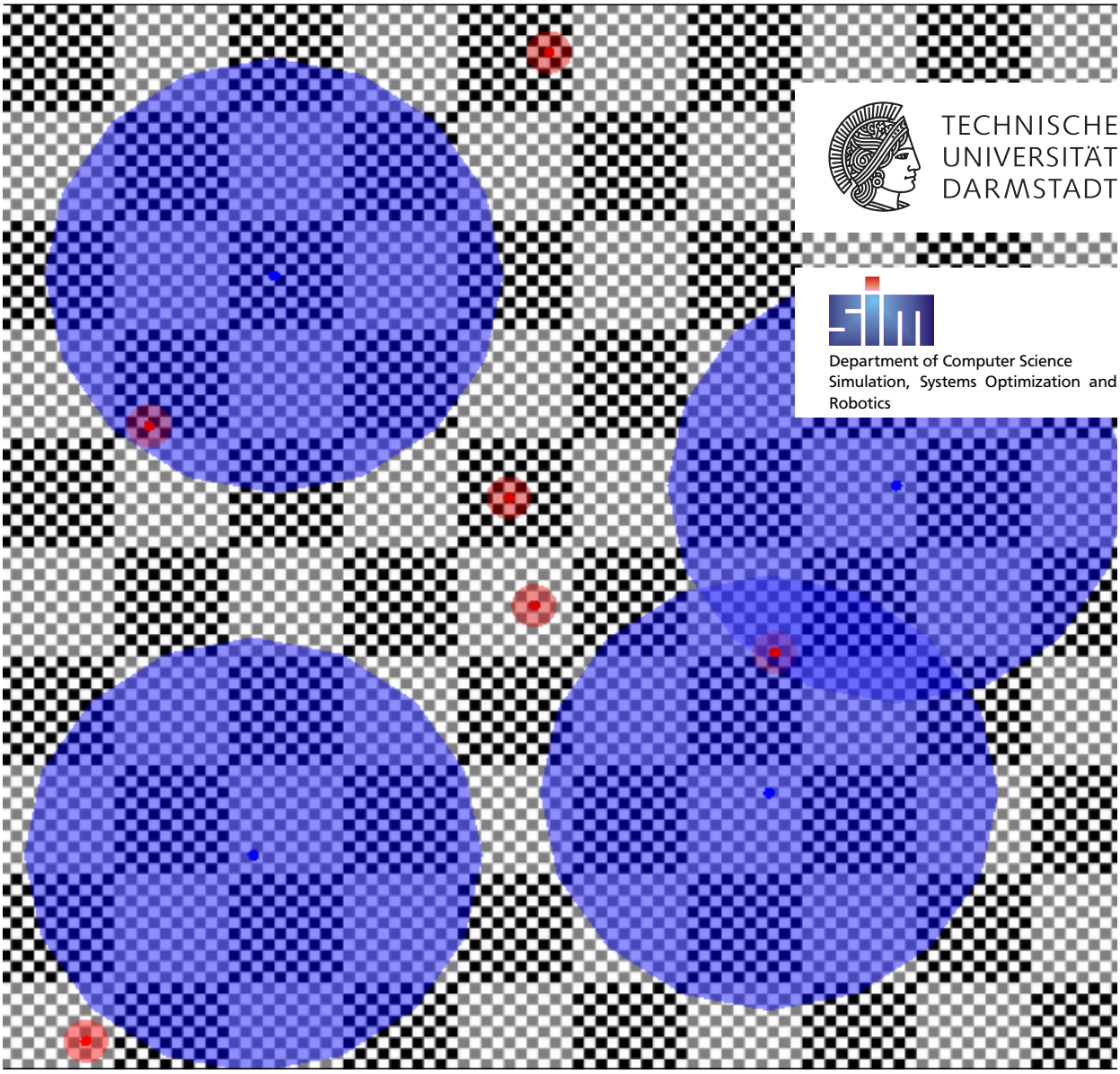
Model-Predictive Control of Cooperative Multi-Vehicle Systems Based on Discrete-Time Linear Systems

Modell-prädiktive Regelung kooperativer Mehrfahrzeugsysteme basierend auf zeit-diskreten linearen Systemen

Diplomarbeit von Juliane Kuhn

Dezember 2009



TECHNISCHE UNIVERSITÄT DARMSTADT

Department of Computer Science
Simulation, Systems Optimization and Robotics

Model-Predictive Control of Cooperative Multi-Vehicle Systems Based on Discrete-Time Linear Systems
Modell-prädiktive Regelung kooperativer Mehrfahrzeugsysteme basierend auf zeit-diskreten linearen Systemen

vorgelegte Diplomarbeit von Juliane Kuhn

1. Gutachten: Prof. Dr. Oskar von Stryk
2. Gutachten: Prof. Dr. Stefan Ulbrich
Betreuer: Dipl.-Math. Christian Reinl

Tag der Einreichung:

Erklärung zur Diplomarbeit

Hiermit versichere ich die vorliegende Diplomarbeit ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 14. Dezember 2009

_____

(Juliane Kuhn)

Abstract

Optimal control of cooperative multi-vehicle systems requires goal-oriented task allocation and simultaneously has to consider vehicle specific motion dynamics. The tight coupling of discrete decision logic and continuous dynamics leads to a high-dimensional hybrid optimal control problem. Solution estimates can be obtained efficiently through discrete-time linear approximations of the physical system behaviour.

Model-predictive control (MPC) based on Mixed-Integer Linear Programs (MILP) is able to allow for the coupling of discrete-continuous structures based on approximated motion dynamics. MPC, in general, combines optimality with robust stabilizing control.

In this thesis, existing MILP-based MPC appoaches are investigated with respect to their suitability for the control of cooperative multi-vehicle systems in real-time applications. Based on a well accepted benchmark scenario, their potential to provide better solutions than existing heuristic approaches is demonstrated. Moreover, the MPC strategy proves to be applicable in real-time, at least for small systems. The presented investigations can serve as a basis for further analysis and development of efficient model-predictive control of cooperative mobility.

Zusammenfassung

Optimale Regelung kooperativer Mehrfahrzeugsysteme erfordert neben zielorientierter Rollenzuweisung die Berücksichtigung fahrzeugspezifischer Bewegungsdynamik. Die enge Kopplung von diskreter Entscheidungslogik und kontinuierlicher Dynamik führt auf ein hochdimensionales hybidres Optimalsteuerungsproblem. Durch eine zeit-diskrete lineare Approximation des physikalischen Systemverhaltens können Näherungslösungen effizient berechnet werden.

Die modell-prädiktive Regelung (eng. model-predictive control (MPC)) basierend auf gemischtganzzahligen linearen Optimierungsprogrammen (eng. mixed-integer linear programs (MILP)) ist in der Lage, die Kopplung diskret-kontinuierlicher Strukturen auf Basis der Dynamikapproximation zu berücksichtigen. MPC im Allgemeinen kombiniert Optimalität mit robuster und stabilisierender Regelung.

In dieser Diplomarbeit werden bestehende MILP-basierte MPC-Ansätze hinsichtlich ihrer Anwendbarkeit für die Regelung kooperativer Mehrfahrzeugsysteme in Echtzeitanwendungen untersucht. Anhand eines anerkannten Benchmark Szenarios wird aufgezeigt, dass die Ansätze das Potential bergen, bessere Lösungen als bestehende heuristische Herangehensweisen zu liefern. Außerdem erweist sich die MPC-Strategie, zumindest für kleine Systeme, als echtzeitfähig. Die vorgestellten Untersuchungen liefern eine Basis für weiterführende Analysen und die Weiterentwicklung effizienter modell-prädiktiver Regler für kooperative Mehrfahrzeugsysteme.

Contents

1 Introduction

1.1 Control of Cooperative Multi-Vehicle Systems

Cooperative behaviour of multiple autonomous vehicles plays an important role in numerous applications. Recent research in this area includes the automation of transportation systems, object surveillance, environment exploration and formation flight [Mur07]. The performance of all these tasks depends on the joint actions or locations of the involved vehicles, i.e. they have to act *cooperatively*.

Cooperative control is comprised of the determination of vehicle-specific trajectories and the situation-based allocation of individual roles and subtasks to the team members. Most applications require robust online control strategies which allow real-time adaptation to a dynamic environment. In this context, one of the biggest challenges is to deal with the complexity of problems involving a large number of interacting vehicles.

There exist approaches, in which actions are allocated based on heuristic methods or behaviour-based decision rules. On the other hand, there are optimization-based approaches which provide optimal control strategies for each vehicle individually or even the whole team at once. Optimization-based methods can provide some optimal solution for every feasible systems state including those a heuristic approach might not cover. On the basis of the underlying model and an accurately designed cost function, it can be guaranteed that the obtained solution is the best possible for the given system state.

Modeling a cooperative control problem must allow for continuous (trajectories) as well as discrete variables and logical rules (allocation of roles). In general, this results in a nonlinear *Hybrid Optimal Control Problem* (HOCP). Solving this problem is computationally very expensive. A discrete-time linear model can very much simplify the HOCP and still provide good solution estimates. Therefore, many optimization-based control strategies use *Mixed Integer Linear Programs* (MILP) to approximate the real system under consideration. The solution of a MILP can be computed efficiently and provides a sequence of optimal control inputs over a certain number of time steps.

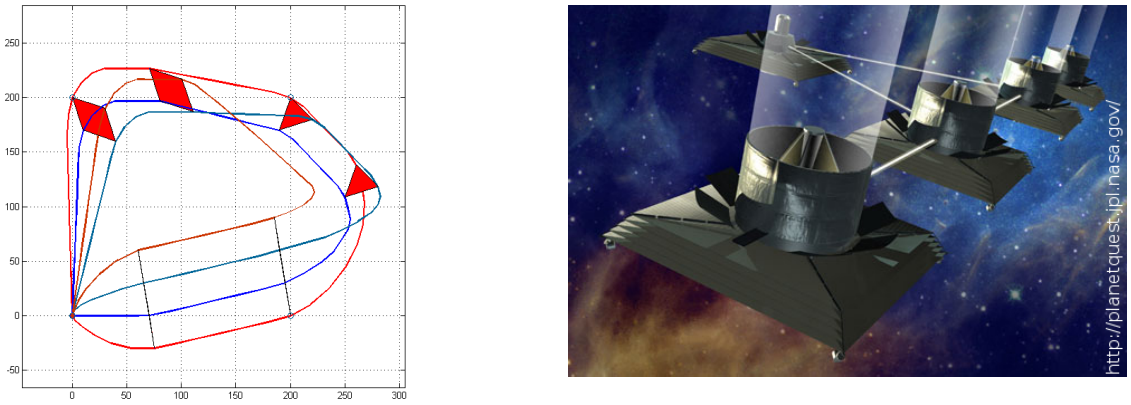
In this thesis, a *model-predictive control* (MPC) strategy for discrete-time hybrid linear systems is investigated. The approach is based on solving multiple MILP in a receding horizon fashion. A stabilizing optimal closed-loop control strategy is obtained. The applicability of this approach to the control of cooperative multi-vehicle systems as well as efficiency and scalability is analysed based on its performance in a representative benchmark scenario.

1.2 Benchmark Scenarios

Formation Control

Formation control is one of the fundamental problems in cooperative control. It can be defined as the *"coordination of a group of robots to follow a given path and to maintain a desired spatial formation"* [KZ08]. Formation control has various applications, for example search and rescue missions, terrain and space exploration, or security patrols to only name a few. Different control strategies have been developed which can roughly be divided into leader-following approaches, virtual structure approaches and behaviour-based appoaches. They comprise centralized as well as decentralized heuristic and optimization-based methods.

A major part of applications concerns unmanned aerial vehicles (UAV), which are to keep a certain formation in order to perform a cooperative operation. An illustrative example of spacecraft applications is the laser inferometer consisting of multiple satellite units flying in formation as depicted in figure 1.1.



(a) (b)

Figure 1.1: (a) Four vehicles cooperatively changing formations [Bar09]; (b) Laser interferometer made of several units flying in formation.

Multiple robots maintaining a formation can also be seen as prerequisite for tasks like cooperative box pushing or cooperative load transportation. Altogether, the problem serves as a rich scalable testbed for control of cooperative multi-vehicle systems.

Robot Games

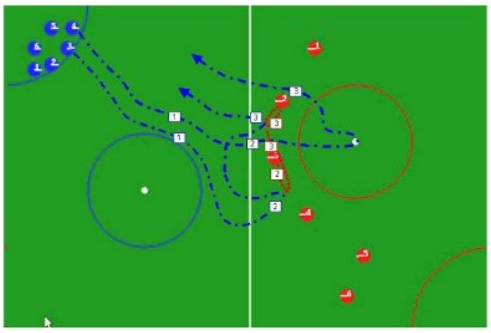
Robot games also provide a variety of benchmark scenarios for control of autonomous robots. Figure 1.2 shows two examples: robot soccer and RoboFlag. In RoboCup Soccer diverse kinds of robots compete in five different leagues with different levels of autonomy. In the small size league, two teams of five robots each have to be coordinated and controlled. For this purpose, a global vision robot located four meters above the field can be used. Hence, task allocation is performed on the basis of global knowledge of the current game situation.

In contrast to that, the robots in the humanoid league are completely autonomous in localizing themselves, communicating with team-mates and percepting the ball, other players, and the field.

The human-like design of the robots poses an additional challenge for the control of dynamical movements.

There is the allocation of tasks like dribbling, defense or kicking the ball on the one hand and the allowance for different physical capabilities for dynamical movement on the other. For this reason, cooperative control in robot soccer combines discrete and continuous system characteristics.

(a) Humanoid robots playing soccer.



(b) Cooperative move in RoboFlag [CDS+03].

Figure 1.2: Robot games as test-bed for cooperative control.

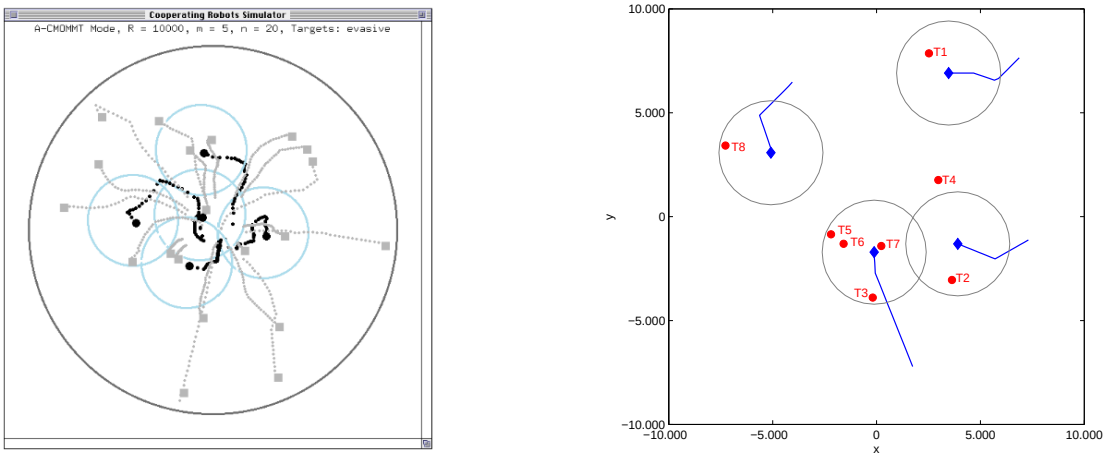
In the RoboFlag game two teams of robots are each trying to capture the other team's flag. The players have to defend opponents from reaching their flag and at the same time try to enter the opposing team's territory. Tasks switch between defense and offense, obstacles on the field have to be avoided and the robots have to communicate relevant information to their team-mates. Hence, the RoboFlag scenario is another possible test-bed for different cooperative control strategies, in particular methods that can efficiently deal with a large number of robots (6-10 per team).

Target Observation

There are various surveillance, security, or rescue scenarios in which multiple targets moving in a bounded area of interest need to be observed. The automation of this task is based on optimal sensor placement to keep all targets in view. In cases where the sensor range and/or the number of available sensors is limited, the sensors are required to dynamically move around the area following the targets to be observed. A successful task performance strongly depends on the robots' cooperation, it cannot be decomposed into independent subtasks. The robots are coupled in their common objective.

In this thesis, a target observation scenario in which multiple targets are to be observed by multiple mobile robots with a 360° field of view with limited range is considered, see figure 1.3 for an example. Movements are restricted to a rectangular bounded area in the plane with no obstacles. Since emphasis is put on the aspect of cooperation, merely simple dynamics are used, that allow robots and targets to move in x- and y-direction and vary their speed. A robot is said to *observe* a target if the target is located within the robot's observation range.

All locations of robots and targets are assumed to be known at all times. This permits a central coordination of all robots. The goal is to minimize the total time in which targets escape observation.



(a) Simulation with moving targets

(b) Simulation with stationary targets

Figure 1.3: Examples of different target observation scenarios: (a) Cooperative Multi-Robot Observation of Multiple Moving Targets [Par02]; (b) Target Observation Problem.

Since the problem involves mobile targets, an online control strategy is needed which evolves dynamically and reacts on changes in the system. Moreover, an optimal control method has to allow for individual physical/dynamical capabilities of the robots. The complexity of the overall problem rises exponentially with the number of involved robots and targets.

Therefore, the considered target observation scenario, in the following referred to as *Target Observation Problem*, is an appropriate scalable test-bed for investigating the applicability of model-predictive approaches to the control of cooperative behaviour in multi-vehicle systems.

1.3 Outline

The rest of the thesis is organized as follows. Chapter 2 gives an overview of related research in the field of cooperative control of multi-vehicle systems. Centralized and decentralized approaches will be reviewed before the model-predictive approach investigated in this thesis is classified among them.

In chapter 3, the basic characteristics of Mixed Logical Dynamical Systems are summarized. Moreover, the concepts of model-predictive control in general and in combination with multi-parametric programming are described. The chapter ends with a short overview of the functionality provided by the Multi-Parametric Toolbox for Matlab.

The model of the Target Observation Problem is derived in chapter 4. Examples of alternative ways of modeling are discussed and indicate the process of developing the final model version. Finally, a possibility to graphically evaluate stability and optimality of problem solutions is presented.

Chapter 5 describes the results of the experiments that were performed to evaluate the solutions obtained from the model-predictive control approach.

The thesis ends with an outlook for real-time applications of the model-predictive controller followed by a final conclusion in chapter 6.

2 Related Research in Control of Cooperative Multi-Vehicle Systems

Cooperative control in general deals with a collection or team of multiple vehicles all seeking to perform a shared task. The task performance depends on the joint actions and mobility of all agents and therefore a complete decomposition of the problem into independent subtasks is not possible [Mur07].

Many different approaches have been proposed for cooperative control problems. They can roughly be divided into centralized and decentralized methods. While *centralized* control considers the cooperative multi-vehicle system as a whole, *decentralized* methods decompose it into coupled subsystems. The decomposition can be hierarchical, dividing the overall problem into different interacting levels. In most cases, however, "decentralized" refers to systems with dynamically independent vehicles coupled by a common objective or cost function.

The following sections provide an overview of different centralized and decentralized approaches to cooperative target observation and related tasks. A more general survey of research in cooperative control is given in [Mur07]. In addition, [NSH04] summarizes aspects of multi-agent model-predictive control in particular, while [Par08] overviews the field of distributed intelligence and task allocation in multi-robot systems.

2.1 Decentralized Approaches

In decentralized control each vehicle individually chooses its next action based on locally available information. This may include locations and (predicted) actions of nearby team-members, called *neighbors*, and parts of the information they gathered. Neighbors share certain bits of information based on some kind of communication protocol. In this context, a way of synchronizing the team members has to be determined.

Since the global problem is divided into smaller subproblems, the computation of decentralized algorithms can be done very efficiently and therefore is suitable for online applications. However, these strategies are often based on heuristics and their solution may be inferior to centralized optimization-based solutions. Approaches, in which each vehicle's next move is based on the solution of some optimization problem, can still find local optima only.

Another way to cope with the key problem of task allocation is the introduction of so called assignment protocols. In this context, [Par98] proposes the behaviour-based architecture ALLIANCE for the control of multi-robot cooperation. Parker's approach uses motivations like impatience and acquiescence for the activation of certain robot behaviour and is able to compensate robot failure or other changes of the environment. ALLIANCE is rather intended for distributed cooperative tasks that can be decomposed into independent or loosely coupled subtasks as for the here investigated class of problems.

Nevertheless, it was also Parker who worked on control strategies for what she calls *inherently* cooperative tasks, i.e. tasks where "*the utility of the action of one robot is dependent upon the current actions of the other team members*" [PT00]. She introduced the problem of *Cooperative Multi-Robot Observation of Multiple Moving Targets* (CMOMMT), which is very similar to the Target Observation Problem, as testbed for inherently cooperative control. In the CMOMMT scenario no global information about the target locations is available. For this reason, the robots must first track down the targets to observe and exchange their local information with nearby team members. Hence, their sensing capabilities include not only a vision range, but also a communication range and a prediction range, which enables the robots to plan their moves based on predicted target movements.
[Par02] describes an approach to the CMOMMT problem that is based on the calculation and weighting of local force vectors, which repel robots from other team members and attract them to nearby targets. Some of the numerical results obtained with this method will be used to evaluate the results of this work in chapter 5.

Multi-robot learning is another approach to deal with cooperative control problems. The state of the art in this branch of research is reviewed in [PL05]. A survey of machine learning methods that are being applied to distributed multi-agent systems is provided among the description of different techniques of team learning and an overview of possible (real world) applications. In [PT00] multi-robot learning is applied to the CMOMMT scenario. An instance-based learning technique is proposed that uses a randomly built lookup table of situation-action pairs to choose the best action according to the current situation. Actions are ranked either by the highest expected reward or a value describing the utility of following actions. Good results were obtained, but could not excel those achieved by a hand-generated solution similar to the one described in [Par02].

The idea of using model-predictive control for cooperative multi-vehicle systems was pursued in [KBB07] and [Dun07]. Both describe distributed MPC versions, where each agent solves an individual optimization problem which includes information on its neighbors and their actions. For both approaches stability could be guaranteed. [KBB07] furthermore provides a detailed look at stability and feasibility issues for distributed MPC.

2.2 Centralized Approaches

Centralized approaches take all available information about the current system state into account and can therefore provide a global control strategy for a whole team of vehicles. Usually this strategy is obtained by solving an optimization problem. Hence, as opposed to decentralized methods, global optimality of the resulting cooperative behaviour can be guaranteed. However, the strategy requires stable communication between the central controller and all of the involved vehicles. Malfunctions in the communication network or vehicle failures are hard to compensate. Since they completely depend on the global controller, the individual vehicles are less autonomous than in decentralized control.

Efficient solvers for (mixed-integer) linear programs are available, which is why (MI)LP are often used to provide solution estimates for nonlinear control problems. In [RvS07b] a transformation of a HOCP representing a heterogeneous cooperative exploration task into a MILP is presented. The resulting system model includes vehicle-specific dynamics and collision avoidance as well as

communication constraints. A similar approach was also applied to a benchmark scenario from robot soccer in [RvS07a].

Depending on the number of involved vehicles and possible actions, considering the system as a whole can result in large and complex optimization problems, the solution of which involves huge computational effort. This can limit the application of optimization-based centralized control to smaller systems. [ED05] proposes a MILP-based approach for RoboFlag drills. A *Mixed Integer Logical Dynamical* (MLD) model is converted into a MILP formulation which is then solved by CPLEX [ILO07]. For large numbers of robots involved in the drill, the required computation times impede using the approach for real-time applications. The authors suggest to use more powerful computers or to perform parts of the computation offline in a multi-parametric fashion.

In order to overcome the issue of computational effort, [CS05] suggests a model simplification by transforming a MILP into an LP which can be used to compute suboptimal solutions of the original MILP. A receding horizon algorithm based on the suboptimal solution then provides the path planning. The approach was applied to different RoboFlag scenarios. The presented method is based on a discretization of the state space, though, and hence is no option for cooperative tasks in which the vehicle dynamics play a decisive role.

2.3 Combined Centralized/Decentralized Approaches

In order to exploit the advantages of both strategies, centralized and decentralized control can be combined. [BF06] proposes a hierarchical combination of centralized and decentralized MILP-based control for a target observation scenario. It comprises a higher-level central optimization algorithm which provides the allocation of targets to the robots. Dynamics as well as obstacle and collision avoidance, respectively, are treated in lower-level MPC-based algorithms by each robot individually. This way, the number of constraints and variables in each optimization problem is reduced and permits an online application of the method.

In order to compare the performance of decentralized versus centralized methods for cooperative target observation, [LSPB05] proposes two (meta)heuristic tunably decentralized algorithms based on hill-climbing and K-means, respectively. The considered problem definition is similar to the Target Observation Problem, but involves randomly moving targets that always outnumber the team of robots. The diverse results that were obtained do not permit a general conclusion whether to prefer centralized or decentralized approaches to the considered scenario.

2.4 Classification and Aim of this Thesis

The model-predictive control approach investigated in this thesis combines the global optimality of a centralized method with efficient closed-loop control. On the basis of a MILP formulation of the problem, the obtained optimal controller directly incorporates motion dynamics with optimal task allocation. However, using MILPs poses a trade-off between linearly approximating the vehicles' physical capabilities on the one hand and the efficiency of available solvers on the other hand.

By using a combination of MPC and multi-parametric programming, the complete optimization procedure could a priori be performed offline. The online control would then reduce to a simple lookup to find the optimal control input for the current system state. Hence, fast online access to the current global optimum would be possible. However, this strategy, first proposed in [BBM00], has so far not been applied to the class of cooperative multi-vehicle scenarios investigated in this thesis. In [Bao05] the multi-parametric MPC approach has successfully been tested for a multi-object adaptive cruise control, [Bor03] describes its application to automotive traction control.

In this thesis, the computational performance of the basic as well as the multi-parametric MPC approach and their suitability for real-time control of complex cooperative multi-vehicle systems is fathomed on the basis of the target observation benchmark scenario introduced in section 1.2.

3 Model-Predictive Control of Hybrid Dynamical Systems

3.1 Modeling Hybrid Systems - The Mixed Logical Dynamical Framework

The *Mixed Logical Dynamical (MLD) Framework* was first proposed in [BM99] for modeling and controlling constrained linear discrete-time systems containing interacting physical laws and logical rules. The latter are transformed into linear inequalities which involve continuous as well as integer variables and thus link dynamics and logic. Common translation techniques (e.g. the Big-M method) are summarized in [BM99].
A system with linear dynamic equations subject to mixed-integer inequalities is obtained:

$$x(k+1) = Ax(k) + B_1 u(k) + B_2 \delta(k) + B_3 z(k) \tag{3.1a}$$

$$y(k) = Cx(k) + D_1 u(k) + D_2 \delta(k) + D_3 z(k) \tag{3.1b}$$

$$E_2 \delta(k) + E_3 z(k) \leq E_1 u(k) + E_4 x(k) + E_5 \ , \tag{3.1c}$$

where $k \in \mathbb{Z}$ the current time step,
$x = [x_c \ x_b]^T, x_c \in \mathbb{R}^{n_c}, x_b \in \{0,1\}^{n_b}$ describes the system state,
$y = [y_c \ y_b]^T, y_c \in \mathbb{R}^{p_c}, y_b \in \{0,1\}^{p_b}$ is the output vector,
$u = [u_c \ u_b]^T, u_c \in \mathbb{R}^{m_c}, u_b \in \{0,1\}^{m_b}$ is the control input,
and $\delta \in \{0,1\}^{r_b}$ and $z \in \mathbb{R}^{r_c}$ represent auxiliary binary and continuous vectors, respectively.

Here only time-invariant systems will be considered, but the MLD framework and all results stated in the following can also be extended to the time-variant case. The definitions presented in this section are based on [BM99].

In general, the inequalities (3.1c) can be satisfied for different values of z and δ. However, a unique dependence of $x(k+1)$ and $y(k)$ on $x(k)$ and $u(k)$ is desirable, which motivates the following definition.

Definition 1 *Let $\mathcal{I}_B$ denote the set of all indices $i \in \{1, \dots, r_b\}$, such that $[B_2]^i \neq 0$, where $[B_2]^i$ denotes the ith column of B_2. Let $\mathcal{I}_D$, $\mathcal{J}_B$, $\mathcal{J}_D$ be defined analogously by collecting the positions of nonzero columns of D_2, B_3, and D_3 respectively. Let $\mathcal{I} \triangleq \mathcal{I}_B \cup \mathcal{I}_D$, $\mathcal{J} \triangleq \mathcal{J}_B \cup \mathcal{J}_D$. A MLD system* (3.1) *is said to be* well posed *if, $\forall k \in \mathbb{Z}$,*

(i) $x(k)$ and $u(k)$ satisfy (3.1c) *for some $\delta(k) \in \{0,1\}^{r_b}$, $z(k) \in \mathbb{R}^{r_c}$, and $x_l(k+1) \in \{0,1\}^{n_l}$;*

(ii) $\forall i \in \mathcal{I}$ there exists a mapping $\mathcal{D}_i : \mathbb{R}^{n+m} \to \{0,1\}$ such that the ith component $\delta_i(k) = \mathcal{D}_i(x(k), u(k))$, and $\forall j \in \mathcal{J}$ there exists a mapping $\mathcal{Z}_i : \mathbb{R}^{n+m} \to \mathbb{R}$ such that $z_j(k) = \mathcal{Z}_j(x(k), u(k))$.

Remark 1 *A MLD system is said to be* completely well posed *if in addition* $\mathcal{I} = \{1, \ldots, r_l\}$ *and* $\mathcal{J} = \{1, \ldots, r_c\}$. *In other words this means that for all* x *and* u *the auxiliary variables* δ *and* z *are uniquely determined and therefore* $x(k+1)$ *and* $y(k)$ *are uniquely defined by* $x(k)$ *and* $u(k)$ [BFTM00].

From the control theoretic point of view, stability of a system is an important issue. Standard stability definitions can be adjusted to fit into the MLD framework.

Definition 2 *A vector* $x_e \in \mathbb{R}^{n_c} \times \{0,1\}^{n_b}$ *is said to be an* equilibrium state *for* (3.1) *and input* $u_e \in \mathbb{R}^{m_c} \times \{0,1\}^{m_b}$ *if* $\left(\begin{smallmatrix} x_e \\ u_e \end{smallmatrix}\right) \in \mathbb{X} \times \mathbb{U}$ *and* $x(k, k_0, x_e, u_e) = x_e$, $\forall t \geq k_0$, $\forall k_0 \in \mathbb{Z}$.
The pair (x_e, u_e) *is said to be an* equilibrium pair.

Definition 3 *Given an equilibrium pair* (x_e, u_e), $x_e \in \mathbb{R}^{n_c} \times \{0,1\}^{n_b}$ *is said to be* stable *if, given* $k_0 \in \mathbb{Z}$, $\forall \varepsilon > 0 \;\; \exists\, \mu(\varepsilon, k_0)$ *such that* $\|x_0 - x_e\| \leq \mu \;\Rightarrow\; \|x(k, k_0, x_0, u_e) - x_e\| \leq \varepsilon$, $\forall k \geq k_0$.

Definition 4 *Given an equilibrium pair* (x_e, u_e), $x_e \in \mathbb{R}^{n_c} \times \{0,1\}^{n_b}$ *is said to be* asymptotically stable *if* x_e *is stable and* $\exists r > 0$ *such that* $\forall x_0 \in \mathcal{B}(x_e, r)$ *and* $\forall \varepsilon > 0 \;\; \exists K(\varepsilon, k_0)$ *such that* $\|x(k, k_0, x_0, u_e) - x_e\| \leq \varepsilon$, $\forall k \geq K$.

For the binary component x_b of the state vector, definition 4 means that there exists a finite time $\bar{k}$ such that $x_b(k) \equiv x_{be}$, $\forall k \geq \bar{k}$. That permits to consider *local stability* only to depend on the continuous part x_c of the state vector. In particular, there exists a neighborhood of x_{ce} in which $x_c(k)$ can be perturbed without violating $x_b(k) = x_{be}$.

Given an equilibrium pair (x_e, u_e), the corresponding values of well posed components of the auxiliary vectors z and δ can be determined via the functions $\mathcal{Z}_i$ and $\mathcal{D}_i$ introduced in definition 1. In order to allow for *indefinite*, i.e. not well posed, components, as well, the following definition is useful.

Definition 5 *Let* (x_e, u_e) *be an equilibrium pair for a time-invariant MLD system, and let the system be well posed by definition 1. For* $i \in \mathcal{I}$, $j \in \mathcal{J}$, *let* $\delta_{e,i}$, $z_{e,j}$ *be the corresponding equilibrium auxiliary variables. An auxiliary vector* δ *(or* z*) is said to be* definitely admissible *if* $\delta_i = \delta_{e,i}$, $\forall i \in \mathcal{I}$, $(z_j = z_{e,j}$, $\forall j \in \mathcal{J})$, *and*

$$E_2\delta + E_3 z \leq E_1 u_e + E_4 x_e + E_5 \,. \tag{3.2}$$

Although the framework was originally developed for hybrid systems, the MLD system class also comprises other important system classes like finite state machines and automata, (constrained) linear systems or nonlinear dynamic systems, where the nonlinearity can be expressed through combinatorial logic, [BM99]. Hence, the MLD framework provides a powerful tool for a wide range of modeling and control tasks, especially when combined with a predictive feedback control strategy.

3.2 Basic Concept of Model Predictive Control

Model Predictive Control (MPC), equivalently referred to as *Receding Horizon Control* (RHC), is an optimal control strategy for (nonlinear) constrained systems. This and the following section illustrate MPC concepts for the control of discrete-time systems, in particular. The basic idea is to use a model of the system and at each sampling time solve an optimization problem, which predicts the optimal state evolution over a finite time horizon N according to some optimality criterion. The output of the optimization procedure is a sequence of optimal control inputs $u(0), \ldots, u(N-1)$. The first element of the sequence is then applied to the system and its state at the next sampling time is measured. The procedure is repeated with the new system state, see figure 3.1. In this manner, the prediction horizon N is shifted or *receded* over time.

Notation: In the following, $x_0, \ldots, x_t, \ldots, x_T$ and $u_0, \ldots, u_t, \ldots, u_{T-1}$ denote the states and inputs at the global time steps t, while $x(0), \ldots, x(k), \ldots, x(N)$ and $u(0), \ldots, u(k), \ldots, u(N-1)$ represent the states and inputs within the scope of the optimization problem, i.e. $x(0) = x_t$, $x(k) = x_{t+k}$, and $u(0) = u_t$, $u(k) = u_{t+k}$.

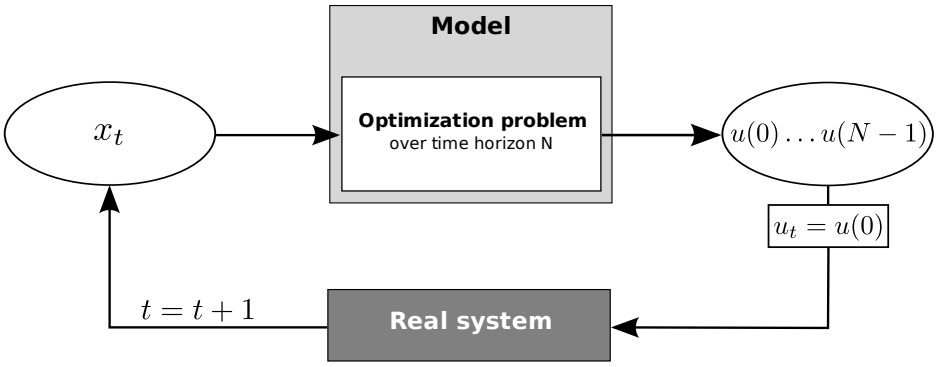


Figure 3.1: Basic idea of MPC

The most obvious advantage of this strategy is the ability to compensate modeling errors or external disturbances of the system. It has been successfully applied to systems with large sampling times, e.g. chemical processes, where the time-span for solving the optimization problem is not limited [Chr06]. Compared to solving an overall optimization problem for all T time steps, the optimization procedure in receding horizon fashion clearly is less complex. Assuming an exponential dependence on the number of time steps, the solution of one optimization problem over the time horizon T would have complexity 2^T, while the model-predictive approach would have complexity $T2^N$. For short prediction horizons N, this is an important advantage over open-loop optimal control strategies.

In the case of hybrid system control, a *Constrained Finite-Time Optimal Control* (CFTOC) problem of the general form

$$\min_{U_N} \quad \|Px(N)\|_p + \sum_{k=0}^{N-1} \|Qx(k)\|_p + \|Ru(k)\|_p \tag{3.3a}$$

$$\text{s.t.} \quad \left.\begin{array}{rcl} x(k+1) & = & A_i x(k) + B_i u(k) + f_i \\ y(k) & = & C_i x(k) + D_i u(k) + g_i \end{array}\right\} \quad \text{if} \begin{pmatrix} x(k) \\ u(k) \end{pmatrix} \in \mathcal{D}_i \tag{3.3b}$$

$$x(N) \in \mathcal{X}_f \tag{3.3c}$$

where $p \in \{1, 2, \infty\}$ is solved at every time step. $U_N := \{u(k)\}_{k=0}^{N-1}$ is the optimization variable and $\mathcal{X}_f$ is a compact terminal target set.

The state update and system output (3.3b) is given as *piecewise affine* (PWA) system over the set of feasible states and inputs. The index i indicates the ith segment of a polyhedral partition $\mathcal{D}$ of the state-input space. A segment is defined by constraints on the x and u variables, called *guard lines*. These are described by inequalities

$$G_i^x x(k) + G_i^u u(k) \leq G_i^c \ . \tag{3.4}$$

The ith dynamics given by (3.3b) will be active if the current state-input combination satisfies (3.4).

Every PWA system description can be transformed into a MLD system description and vice versa. A proof of equivalence can be found in [BFTM00]. Therefore, all theoretical results mentioned in section 3.1 also hold for PWA systems. On the other hand, methods developed for PWA systems can be extended to the MLD framework.

One way to solve the CFTOC problem is to use its MLD representation and transform it into a *Mixed Integer Linear/Quadratic Problem* (MILP/MIQP), depending on $p = 1, \infty$ or $p = 2$. For this purpose, problem (3.3) is rewritten as

$$\min_{U_N} \quad \|Px(N)\|_p + \sum_{k=0}^{N-1} \|Q_1 u(k)\|_p + \|Q_2 \delta(k)\|_p + \|Q_3 z(k)\|_p + \|Q_4 x(k)\|_p \tag{3.5a}$$

$$\begin{aligned} \text{s.t.} \quad & x(k+1) = Ax(k) + B_1 u(k) + B_2 \delta(k) + B_3 z(k) \\ & y(k) = Cx(k) + D_1 u(k) + D_2 \delta(k) + D_3 z(k) \qquad (3.5b) \\ & E_2 \delta(k) + E_3 z(k) \leq E_1 u(k) + E_4 x(k) + E_5 \\ & x(N) \in \mathcal{X}_f \ . \qquad (3.5c) \end{aligned}$$

A standard approach for translating (3.5) into a MILP or MIQP is, among others, described in [Bor03]. For the class of mixed-integer problems efficient (branch-and-bound) solvers like CPLEX [ILO07] exist. However, MILP/MIQP are $\mathcal{NP}$-hard problems. Thus, depending on the model's size and complexity, the time needed to compute the next optimal input in a MPC strategy can become prohibitive for fast systems.

MPC provides feedback control tools to stabilize MLD and PWA systems, respectively, or to drive them to desired reference trajectories. Under certain assumptions, stability can even be guaranteed, as stated in the following theorem.

Theorem 1 [BM99] *Let (x_e, u_e) be an equilibirium pair and let (δ_e, z_e) be definitely admissible. Assume that the initial state x_0 is such that a feasible solution of problem* (3.5) *exists at time $t = 0$. Then $\forall Q_1 \succeq 0$ (i.e. for Q_1 positive semidefinite), $\forall Q_2 \succeq 0$, $\forall Q_3 \succeq 0$, and $\forall Q_4 \succeq 0$ the mixed-integer predictive law* (3.5) *with $u_t = u(0)$ according to the receding horizon strategy stabilizes the system in that*

$$\begin{aligned}
&\lim_{t\to\infty} x_t = x_e \\
&\lim_{t\to\infty} u_t = u_e \\
&\lim_{t\to\infty} \|Q_2(\delta_t - \delta_e)\| = 0 \\
&\lim_{t\to\infty} \|Q_3(z_t - z_e)\| = 0 \,,
\end{aligned}$$

while fulfilling the constraints (3.1c).

In most applications not all matrices Q_i satisfy the requirements of theorem 1 or shall explicitly be left as free tuning parameters. In that case, for non-singular matrices Q_i, stability can be guaranteed by introducing certain constraints on the terminal state x_T or by appropriately choosing the terminal weight P (cf. [BBM00, BBM02, Kva08]).

3.3 Explicit MPC

For fast systems the complexity of the optimization problem is limited due to the required computation time to solve it online. Therefore, only very short prediction horizons can be used or the system model needs to be simplified. In cases where the problem complexity cannot be reduced accordingly, the application of online MPC might not be possible at all [Chr06].

Bemporad et al. proposed a modification of the general MPC concept in which the optimization procedure is a priori performed offline [BBM00, BBM02]. The state vector x_t is considered a free parameter and the optimization problem is once solved for all x_t in the state space. This is done by solving a *multi-parametric* optimization program. More details on multi-parametric programming will be given in the next section.

This procedure provides a lookup table for the optimal control input u_t, as can be derived from the following theorem.

Theorem 2 [Bor03] *The solution to the optimal control problem* (3.3) *with $p = 1, \infty$ is a PWA state feedback control law of the form $u^*(k) = f_k(x(k))$,*

$$f_k(x(k)) = F_k^i x(k) + g_k^i \qquad \text{if} \quad x(k) \in \mathcal{P}_k^i \,,\ k = 0, \ldots, N-1 \tag{3.6}$$

where $\mathcal{P}_k^i$, $i = 1, \ldots, N_k^r$, is a polyhedral partition of the set X_k of feasible states $x(k)$.

By solving the MILP version of the CFTOC in a multi-parametric fashion and as a result of theorem 2, the control input $u(k)$ can be explicitly expressed as a piecewise affine function of the current state $x_t = x(0)$, i.e.

$$u(k) = \mu(x_t, k) \qquad \text{for} \quad k = 0, \ldots, N \,. \tag{3.7}$$

Since for the MPC method only the first element of the optimal control sequence is of interest (3.7) can be simplified to

$$\mu_{RH}(x_t) = \mu(x_t, 0) = \mu_i(x_t) \qquad \text{if} \quad x_t \in \mathcal{P}^i \,, \tag{3.8}$$

where $\mathcal{P}^i := \mathcal{P}^i_0$. (3.8) is called the *optimal lookup table* or *explicit solution* [Chr06, BBM02]. During the online computation the control input at time step t is then given by

$$u_t = \mu_{RH}(x_t) \,. \tag{3.9}$$

Figure 3.2 illustrates the explicit MPC concept.

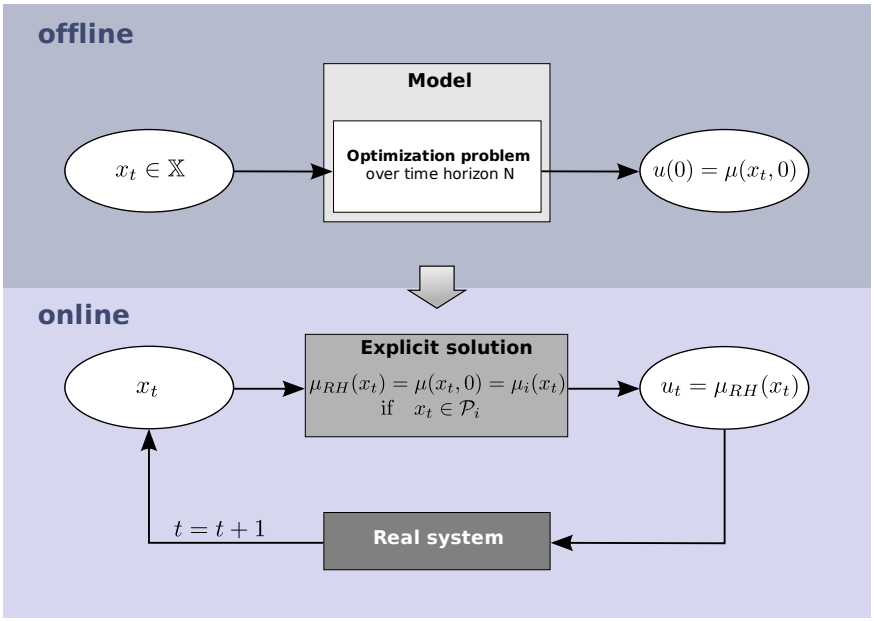


Figure 3.2: Explicit MPC scheme

The evaluation of the control law (3.9) requires the identification of the state space region $\mathcal{P}^i$, such that $x_t \in \mathcal{P}^i$. This leads to the so called *point location problem* for which an efficient solution is described in [Kva08].

Remark 2 *Similar results hold for the CFTOC with quadratic performance index, i.e.* $p = 2$. *As opposed to the linear case, however, the partition of the feasible state space need not be polyhedral* [Bor03].

Remark 3 *The explicit feedback control law* (3.9) *and the control law defined by* (3.3) *and* (3.5)*, respectively, are exactly equal,* [BMDP02]. *Therefore, identical control inputs will be obtained by both the online and the offline version of MPC.*

The main motivation for the explicit MPC is its applicability to systems with high sampling rates. Solving the point location problem is usually very fast compared to solving an optimization problem at every sampling time. Moreover, one can provide an exact upper bound for the duration of the point location search, whereas a corresponding bound for the online optimization procedure is hard to estimate.

Implementing an online model-predictive controller requires a large computational infrastructure. In contrast, the explicit solution, once computed, can easily be implemented and reproduced on other machines or even microprocessors.

The explicit MPC solution allows better understanding of the MPC characteristics like stability, feasibility or robustness. In addition, it provides more visualization options. Along with the analysis capacity comes the potential for post-processing the solution. It may be simplified, its complexity may be reduced, or the solution can be adapted in other ways to meet individual requirements.

On the other hand, multi-parametric optimization problems are difficult to solve and the required computation time in the worst case depends exponentially on the problem size, in particular on the number of integer variables involved [BM99]. This limits the applicability of the explicit MPC method to problems with a low number of constraints and binary variables and a short prediction horizon. The problem dimension also influences the solution complexity, i.e. the number of regions $\mathcal{P}^i$ to compute and the storage demand of the resulting lookup table.

3.4 Multi-Parametric Programming

Parametric programming is used to obtain the optimal solution to some problem for a whole range of different parameters. Considering the CFTOC problem (3.3), a characterization of the optimal control input u^* for all x in the feasible state space X is desired. Since the CFTOC solution depends on a vector of parameters, as opposed to a single scalar, it is referred to as *multi-parametric* program.

The solution of a multi-parametric program provides a partition of the state space into so called *critical regions* CR_i, i.e. "sets of the parameter space where the local conditions for optimality remain unchanged" [Bor03]. More precisely, inside a critical region CR_i the combination of active constraints at the optimal solution does not change. An explicit dependence of the optimizer u^* on the parameters $x \in CR_i$ is obtained.

In order to illustrate the partitioning process, consider a non-degenerate multi-parametric linear program (mpLP). First, the smallest affine subspace $\mathcal{K}$ that contains the feasible state space X of the LP has to be determined (standard procedure, e.g. [Bor03], algorithm 1.3.2). Then an arbitrary state vector $x_0 \in \mathcal{K}$ is chosen to solve the LP in order to obtain the optimizer u^* and the set $A(x_0)$ containing the indices of the corresponding active constraints. $A(x_0)$ defines the first critical region $CR_0 = CR_{A(x_o)} = \{x \in X : A(x) = A(x_0)\}$. An effective way of partitioning the rest of the space $CR^{rest} = X \setminus CR_0$ is proposed in the following theorem.

Theorem 3 [BMDP02] *Let $Y \subseteq \mathbb{R}^n$ be a polyhedron, and $CR_0 \triangleq \{x \in Y : Ax \leq b\}$ a polyhedral subset of Y, $CR_0 \neq \emptyset$. Also let*

$$R_i = \{x \in Y : A^i x > b^i \text{ and } A^j x \leq b^j, \forall j < i\}, \quad i = 1, \ldots, m ,$$

where $m = dim(b)$, and let $CR^{rest} \triangleq \bigcup_{i=1}^{m} R_i$. Then

(i) $CR^{rest} \cup CR_0 = Y$ and

(ii) $CR_0 \cap R_i = \emptyset$, $R_i \cap R_j = \emptyset$, $\forall i \neq j$,

i.e. $\{CR_0, R_1, \ldots, R_m\}$ is a partition of Y.

A two-dimensional example of the procedure described in theorem 3 is shown in figure 3.3. If the problem under consideration is not degenerate, the partition of the feasible state space X is uniquely defined [Bor03].

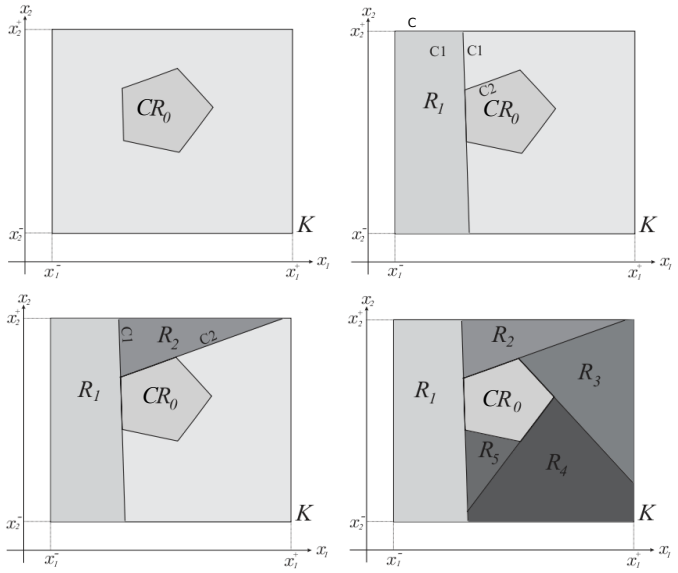
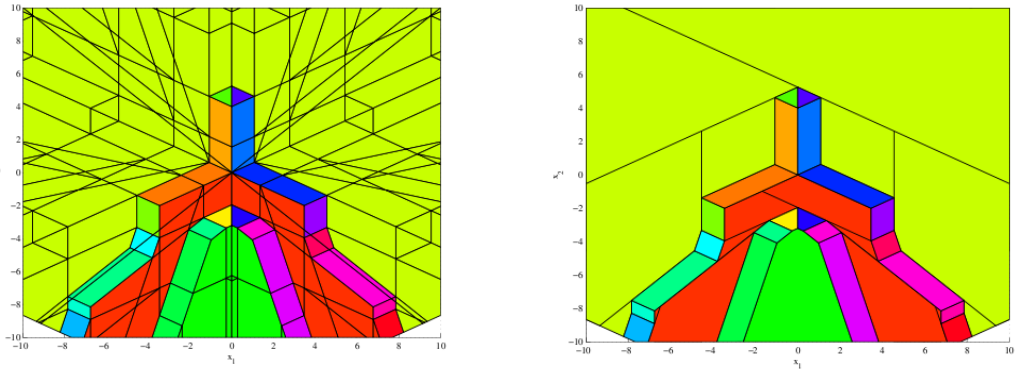


Figure 3.3: Partition of the rest of the space [BMDP02].

Some of the regions in the solution can appear more than once or overlap each other. In order to reduce the complexity of the solution duplicates and overlaps are to be removed. Efficient algorithms to determine a minimal representation of the solution are presented in [Kva08]. Figure 3.4 shows an example of a solution before and after simplification.

[DP00] describes an efficient approach for determining the solution of a multi-parametric *mixed-integer* linear program (mpMILP) based on iteratively solving a MILP and a mpLP subproblem. It follows from theorem 2 that the obtained solution is a piecewise affine function in x defined over a polyhedral partition of the state space. The algorithm can be summarized as follows:

1. In a first step, the given MILP is solved to obtain initial values for the binary variables δ.
2. The binary variables are fixed, i.e. $\bar{\delta} = \delta$, and a mpLP is solved which provides a partition of the state space into smaller critical regions CR_i.
3. For each CR_i a MILP is solved for $x \in CR_i$.
   If a value better than the current objective value can be obtained with a different combination of binary values δ^*, the upper bound for the objective in the region is updated and



(a) Regions of explicit solution before simplification (252 regions)

(b) Regions of explicit solution after simplification (39 regions)

Figure 3.4: Example of a polyhedral partition of the state space [KGBC06].

$\bar{\delta} = \delta^*$ is used as new basis for again solving a mpLP to further partition the current region. If no better feasible solution is found, CR_i is marked infeasible and will not be explored further.

4. The algorithm will terminate when no feasible regions are left.

3.5 The Multi-Parametric Toolbox

The Multi-Parametric Toolbox (MPT) is a free Matlab toolbox that was developed by members of the Automatic Control Laboratory at the ETH Zurich [KGBC06]. It provides novel efficient algorithms for the design and analysis of model predictive controllers for constrained (non)linear and hybrid systems. Its features include modeling of hybrid systems and the import of existing system structures, controller export to C code and simulations in Matlab Simulink, analysis of hybrid systems, multi-parametric optimization as well as visualization tools and a Graphical User Interface (GUI) for the controller design.

The MPT efficiently implements the MPC procedures described in sections 3.2 and 3.3. It offers a huge library of state-of-the-art algorithms from fields like computational geometry and multi-parametric programming [Kva08]. For the explicit solution of CFTOC problems like the one considered in this thesis, multi-parametric programs are solved recursively in a dynamic programming fashion [BBBM03]. The algorithm used for solving the Point Location Problem, i.e. the online evaluation of the resulting explicit control law, is based on a binary search tree as proposed in [TJB03]. See [Kva08] and [KGBC06] for a complete description of the MPT functionality, case studies proving its efficiency, and a comparison to other available software packages like the Hybrid Toolbox [Bem04] or the MPC Toolbox for Matlab.

In this thesis, the MPT functionality and its usability for the design and analysis of model predictive controllers for the Target Observation Problem is explored. Two types of controllers have to be distinguished. The MPT implementation of the basic MPC concept as presented in section 3.2 is in the following referred to as *online controller* while the term *explicit controller* refers to the MPT realisation of the closed-form solution as described in section 3.3.

4 Modeling a Target Observation Scenario

4.1 Constraints

Dynamics

The focus of investigation in this thesis is control of cooperative behaviour. Therefore, it suffices to model simplified vehicle dynamics and consider robots and targets to be point masses moving in the plane. However, the modular structure of the overall system model permits to substitute a simple dynamic model with any other more complex model if necessary.

Let x and y denote the robots' position coordinates, v_x and v_y the corresponding velocities, u_x and u_y the corresponding accelerations. The system model will only include dynamically moving robots. In a first problem version the targets are assumed to stay in predefined positions. Therefore, the model only includes constant target positions and no velocities. Later, also mobile targets will be considered. Since the robots' behaviour is only based on the current target positions, there is no need to extend the model with target dynamics. Their movement will be computed outside the MPC scheme. For slowly moving targets the controller considers the modified target positions as noise on the constant values and will adjust the system state accordingly.

The robot dynamics are described by the differential equations $\ddot{x}(t) = u_x(t)$ and $\ddot{y}(t) = u_y(t)$ which can be transformed into the first order differential equations

$$\begin{aligned} \dot{x}(t) &= v_x & \dot{y}(t) &= v_y(t) \\ \dot{v}_x(t) &= u_x & \dot{v}_y(t) &= u_y(t)\,. \end{aligned}$$

The application of Euler's method then provides discrete-time linear systems of the form

$$\begin{aligned} \begin{pmatrix} x^{(k+1)} \\ v_x^{(k+1)} \end{pmatrix} &= \begin{pmatrix} x^{(k)} \\ v_x^{(k)} \end{pmatrix} + \Delta t \begin{pmatrix} v_x^{(k)} \\ u_x^{(k)} \end{pmatrix} \\ \begin{pmatrix} y^{(k+1)} \\ v_y^{(k+1)} \end{pmatrix} &= \begin{pmatrix} y^{(k)} \\ v_y^{(k)} \end{pmatrix} + \Delta t \begin{pmatrix} v_y^{(k)} \\ u_y^{(k)} \end{pmatrix}, \end{aligned} \tag{4.1}$$

where $k = 1, \ldots, N$, $\Delta t = t_{k+1} - t_k$, $x^{(k)} := x(t_k)$, and $y^{(k)}$, $v_{x,y}^{(k)}$, $u_{x,y}^{(k)}$ defined analogously.

Notation: For reasons of clarity, throughout this section and section 4.4, all variables will be denoted without time reference, e.g. $x \triangleq x^{(k)}$.

Distances

The exact Euclidean distance between a robot r and a target t is given by $d_{rt} = \sqrt{(x_r - x_t)^2 + (y_r - y_t)^2}$. This expression can be linearized using the following approximation:

$$d_{rt} \approx \min \left\{ \tilde{d}_{rt} \;\middle|\; (x_r - x_t)\sin\frac{2\pi j}{n_d} + (y_r - y_t)\cos\frac{2\pi j}{n_d} \le \tilde{d}_{rt},\ j = 1, \ldots, n_d,\ n_d \in \mathbb{N} \right\}. \tag{4.2}$$

Observation Constraints

A target is said to be observed if it is located within some robot's observation range with radius R. Let n_R be the number of robots, n_T the number of targets involved in the problem. For each combination (r,t), where $r \in \{1 \dots n_R\}$ and $t \in \{1 \dots n_T\}$ a binary variable $b_{rt} \in \{0,1\}$ indicates whether or not robot r currently observes target t. This is expressed by the following implication:

$$b_{rt} = 1 \quad \Rightarrow \quad d_{rt} \leq R \,. \tag{4.3}$$

All robots are assumed to have observation ranges of equal size. Applying the Big-M method to (4.3) results in the inequality

$$d_{rt} - R \quad \leq \quad M(1 - b_{rt}) \,, \tag{4.4}$$

where $M \geq \max(d_{rt} - R)$.

Since it is not of interest *which* robot observes target t, but if it is observed by *any* of them, another binary variable $s_t \in \{0,1\}$ is introduced and represents the general observation status of target t:

$$s_t = 0 \quad \Leftrightarrow \quad \sum_r b_{rt} \geq 1 \,. \tag{4.5}$$

A linear formulation of (4.5) is given by the two inequalities

$$\begin{aligned} 1 - \sum_r b_{rt} &\leq M \cdot s_t \quad \text{and} \\ 1 - \sum_r b_{rt} &\geq \epsilon + (m - \epsilon)(1 - s_t) \,, \end{aligned} \tag{4.6}$$

where $M \geq \max(1 - \sum_r b_{rt}) = 1$, $m \leq \min(1 - \sum_r b_{rt}) = 1 - n_R$ and ϵ a small tolerance close to machine precision.

The aspect of cooperation is realised by minimizing the number of unobserved targets, i.e. $\sum_t s_t$, and by minimizing each robot's distance to those targets not yet observed by any other robot. The latter decision can be expressed using the binary variables s_t and an additional set of auxiliary variables $h_{rt} \in \mathbb{R}$, $r \in \{1 \dots n_R\}$, $t \in \{1 \dots n_T\}$ which equal the distances d_{rt} in the case of unobserved targets and equal zero in the case of already observed targets:

$$h_{rt} = s_t \cdot d_{rt} \,. \tag{4.7}$$

The corresponding linear representation comprises the inequalities

$$\begin{aligned} h_{rt} &\leq M \cdot s_t \,, \\ -h_{rt} &\leq -m \cdot s_t \,, \\ h_{rt} &\leq d_{rt} - m(1 - s_t) \,, \quad \text{and} \\ -h_{rt} &\leq -d_{rt} + M(1 - s_t) \,. \end{aligned} \tag{4.8}$$

Here, $M = \max(d_{rt})$ and $m = \min(d_{rt}) = 0$ and hence (4.8) reduces to

$$\begin{aligned} h_{rt} &\leq M \cdot s_t \,, \\ h_{rt} &\leq d_{rt} \,, \quad \text{and} \\ -h_{rt} &\leq -d_{rt} + M(1 - s_t) \,. \end{aligned} \tag{4.9}$$

By minimizing the variables h_{rt} only distances to unobserved targets affect the objective value. Already observed targets are "ignored" by the robots.

4.2 Objectives

Clearly, the main objective of the Target Observation Problem is to observe as many targets as possible for the largest possible period of time, i.e. in the discrete-time case for as many time steps as possible. In order to achieve this goal the robots are to move towards strategically good positions. This means that a robot is supposed to approach unobserved targets without losing sight of the target it might already be observing. This condition becomes even more important when mobile targets are considered and shall ensure that the robot is in an advantageous position to observe additional targets as soon as they enter its observation range. At last, the robots are to move at a minimum control effort, which in reality could correspond to fuel consumption or other limiting factors. In summary, the cost function contains the following elements:

- The number of unobserved targets:

$$\min \sum_{k} \sum_{t} s_t^{(k)} \tag{4.10}$$

- The distances to unobserved targets:

$$\min \sum_{k} \sum_{r,t} h_{rt}^{(k)} \tag{4.11}$$

- The required control effort:

$$\min \sum_{k} \sum_{r} |u_{r,x}^{(k)}| + |u_{r,y}^{(k)}| \tag{4.12}$$

The three elements (4.10) - (4.12) have to be weighted according to the different objective priorities and the best expected task performance. For this purpose, the weights q_δ, q_z, $q_u \in \mathbb{R}$ are introduced and the assembled cost function is of the form

$$\min \quad q_\delta \cdot \sum_{k} \sum_{t} s_t^{(k)} \; + \; q_z \cdot \sum_{k} \sum_{r,t} h_{rt}^{(k)} \; + \; q_u \cdot \sum_{k} \sum_{r} |u_{r,x}^{(k)}| + |u_{r,y}^{(k)}| \; . \tag{4.13}$$

4.3 Linear Problem Formulation

The overall linear discrete-time mixed-integer representation of the Target Observation Problem is composed of the cost function (4.13), the constraints for dynamics (4.1), distances (4.2), observation (4.4), (4.6), and (4.9), as well as the specification of upper and lower bounds for all variables:

$$
(TOP)
$$

$$
\begin{aligned}
\min \quad & q_\delta \cdot \sum_k \sum_t s_t^{(k)} \;+\; q_z \cdot \sum_k \sum_{r,t} h_{rt}^{(k)} \;+\; q_u \cdot \sum_k \sum_r |u_{r,x}^{(k)}| + |u_{r,y}^{(k)}| \\
\\
\text{s.t.} \quad & x_r^{(k+1)} \;=\; x_r^{(k)} + \Delta t\, v_{r,x}^{(k)} \\
& y_r^{(k+1)} \;=\; y_r^{(k)} + \Delta t\, v_{r,y}^{(k)} \\
& v_{r,x}^{(k+1)} \;=\; v_{r,x}^{(k)} + \Delta t u_{r,x}^{(k)} \\
& v_{r,y}^{(k+1)} \;=\; v_{r,y}^{(k)} + \Delta t u_{r,y}^{(k)} \\
\\
& (x_r^{(k)} - x_t^{(k)}) \sin \frac{2\pi j}{n_d} + (y_r^{(k)} - y_t^{(k)}) \cos \frac{2\pi j}{n_d} \;\leq\; d_{rt}^{(k)} \\
\\
& d_{rt}^{(k)} - R \;\leq\; M_1 (1 - b_{rt}^{(k)}) \\
\\
& 1 - \sum_r b_{rt}^{(k)} \;\leq\; M_2 \cdot s_t^{(k)} \\
& 1 - \sum_r b_{rt}^{(k)} \;\geq\; \epsilon + (m - \epsilon)(1 - s_t^{(k)}) \\
\\
& h_{rt}^{(k)} \;\leq\; M_3 \cdot s_t^{(k)} \\
& h_{rt}^{(k)} \;\leq\; d_{rt}^{(k)} \\
& -h_{rt}^{(k)} \;\leq\; -d_{rt}^{(k)} + M_3 (1 - s_t^{(k)}) \\
\\
& u_{min} \;\leq\; u_{r,x}^{(k)}, u_{r,y}^{(k)} \;\leq\; u_{max} \\
& v_{min} \;\leq\; v_{r,x}^{(k)}, v_{r,y}^{(k)} \;\leq\; v_{max} \\
& x_{min} \;\leq\; x_r^{(k)}, x_t^{(k)} \;\leq\; x_{max} \\
& y_{min} \;\leq\; y_r^{(k)}, y_t^{(k)} \;\leq\; y_{max} \\
& b_{rt}^{(k)}, s_t^{(k)} \in \{0, 1\} \\
\\
& \text{where } M_1 \geq \max(d_{rt} - R),\; M_2 \geq \max(1 - \sum_r b_{rt}) = 1,\; M_3 \geq \max(d_{rt}), \\
& m \leq 1 - n_R,\; r = 1, \ldots, n_R,\; t = 1, \ldots, n_T,\; j = 1, \ldots, n_d,\; k = 1, \ldots, N
\end{aligned}
$$

MILP Formulation

Problem (TOP) is a MILP and can be solved directly with any available MILP solver. Its optimal solution can serve as a reference for evaluating the solutions obtained from the MPC approach as well as the controller performance. Moreover, size and complexity of the MILP can be compared to the MLD problem formulation described in the next section. For these reasons, the Target Observation Problem was first formulated as MILP of the general form

$$\begin{aligned} \min_{x} \quad & c^T \begin{pmatrix} x \\ u \end{pmatrix} \\ \text{s.t.} \quad & A_1 \begin{pmatrix} x \\ u \end{pmatrix} = b_1 \\ & A_2 \begin{pmatrix} x \\ u \end{pmatrix} \leq b_2 \ . \end{aligned} \tag{MILP}$$

Here, the vector $\binom{x}{u}$ contains all variables introduced in section 4.1, i.e. a total of $\left(8n_R + n_R n_T\right) T + 2n_T$ continuous and $\left(n_T + n_R n_T\right) T$ binary variables, where T is the considered number of time steps. The entire MILP comprises $10n_R + 2n_T + 8n_R T + n_T T + 2T n_R n_T n_d$ constraints. Its optimal solution $\binom{x^*}{u^*}$ is computed by CPLEX [ILO07] and provides an optimal open-loop control sequence for the problem.

MLD Formulation

In order to apply the concept of MPC and to explore the MPT functionality, problem (TOP) has to be reformulated as CFTOC of the form (3.5) employing the MLD framework (cf. section 3.1):

$$\min_{U_N} \quad |Px(N)| + \sum_{k=0}^{N-1} |Q_1 u(k)| + |Q_2 \delta(k)| + |Q_3 z(k)| + |Q_4 x(k)| \tag{4.14a}$$

$$\text{s.t.} \quad x(k+1) = Ax(k) + B_1 u(k) + B_2 \delta(k) + B_3 z(k) \tag{4.14b}$$

$$y(k) = Cx(k) + D_1 u(k) + D_2 \delta(k) + D_3 z(k) \tag{4.14c}$$

$$E_2 \delta(k) + E_3 z(k) \leq E_1 u(k) + E_4 x(k) + E_5 \tag{4.14d}$$

$$x(N) \in T_{set} \ . \tag{4.14e}$$

In this representation of the problem, the vector $x(k) \in \mathbb{R}^{4n_R+2n_T}$ contains robot positions, robot velocities, and target positions only. All binary variables are contained in the vector $\delta(k) \in \{0,1\}^{n_R \cdot n_T + n_T}$. $z(k) \in \mathbb{R}^{2 \cdot n_R \cdot n_T}$ comprises all other (auxiliary) continuous variables like the distances $d_{rt}(k)$ and $h_{rt}(k)$, respectively. The vector $u(k) \in \mathbb{R}^{2n_R}$ represents the robot control inputs.

(4.14b) consists of $4n_R + 2n_T$ equalities. Since there is no special use for the system output $y(k)$, line (4.14c) was "eliminated", i.e. it does not yield additional constraints. (4.14d) comprises $4n_R n_T + 2n_T + n_R n_T n_d$ inequalities.

The MLD framework was exploited as favorable as possible in terms of problem complexity also taking into account how minimizing the cost function affects the variables. As an example, consider the first observation constraint in section 4.1. A correct representation would require an

"*if-and-only-if*"-relation instead of the single implication (4.3). However, the optimization will always drive the variables b_{rt} to value 1 instead of 0, which is why (4.3) suffices to completely describe the relation between b_{rt} and d_{rt} for problem (4.14). The accuracy of the distance approximation (4.2) itself also depends on the minimization of the variables d_{rt} and h_{rt}, respectively.

The disadvantage of this strategy is that the resulting MLD model (4.14b) - (4.14d) is not completely well posed in the sense of remark 1. Without the influence of optimization, the functions $\mathcal{D}_i$ and $\mathcal{Z}_i$, as introduced in definition 1, do not provide unique values for the components of $\delta(k)$ and $z(k)$, respectively. A completely well posed model would be far more complex since more constraints and more variables, especially more binary variables, would be needed. In terms of the computational time for an explicit controller this was to be avoided. Note that the reduction of the problem formulation does not influence or deteriorate the results of the overall optimization procedure.

4.4 Alternative Modeling Approaches

Despite the reduced problem representation described in the previous section, the explicit solution (cf. section 3.3) of the CFTOC problem (4.14) is still quite complex. Therefore, depending on the number of involved robots and targets and the size of the prediction horizon N, the computation of an explicit controller turned out to be extremely expensive (for details see section 5.1. Since the number of binary variables in the MLD model considerably influences the solution complexity, alternative ways of modeling with a reduced number of binary variables were tested. In the following, an example of an alternative observation constraint will be presented.

On the other hand, a complete characterization of the developed MLD model is desirable in order to analyse it with respect to the theoretical results discussed in section 3.1. One of the key properties is wellposedness of an MLD system. An alternative specification of distances is presented in this section. It intends to provide uniquely determined distances independend of the optimization influence and, hence, could help to overcome one of the disadvantages mentioned in section 4.3.

Alternative Observation Constraint

The approach described in the following intends to reduce the number of binary variables in the MLD model by eliminating the $n_R n_T$ variables b_{rt}. The idea is to check the minimum distance of all robots to target t in order to determine if target t is observed by any of the robots. If the minimum distance is smaller than the vision radius rad, the target is observed.

Let $t \in \{1 \dots T\}$. Then

$$s_t = 0 \quad \Rightarrow \quad \min_{r \in \{1 \dots n_R\}} d_{rt} \leq R. \tag{4.15}$$

The translation of this statement into linear inequalities requires an auxiliary variable $l_t \in \mathbb{R}$ representing the minimum distance. With $l_t = -\min_{r \in \{1 \dots n_R\}} d_{rt}$ (4.15) reduces to

$$s_t = 0 \quad \Rightarrow \quad l_t \geq -R\,, \tag{4.16}$$

which can be transformed into

$$\begin{aligned} l_t &\geq -d_{rt} \,, \qquad r = 1, \ldots, n_R \\ -d_{max} &\leq l_t \leq 0 \,, \\ -l_t - R &\leq M \cdot s_t \,, \end{aligned} \tag{4.17}$$

where $M = \max(-l_t - R) = \max(d_{rt} - R)$.

In order to obtain the correct values for l_t, the cost function has to be modified such that *all* distances d_{rt} as well as the new variables l_t are minimized.

The main problem discovered with this approach is that its accuracy depends too much on the effect of minimizing the cost function. It proves to be impossible to accordingly weight the sums in the cost function and obtain useful approximations of d_{rt} and l_t. For this reason, the approach is no longer pursued.

Alternative Specification of Distances

This modification of the distance model aims to obtain uniquely determined values for the variables d_{rt} and, hence, come one step closer to a completely well posed MLD model.

When using the "Manhattan metric" instead of the Euclidean metric the distance d_{rt} between robot r and target t is given by

$$d_{rt} = |x_r - x_t| + |y_r - y_t| \,. \tag{4.18}$$

The absolute value of a function $g(x) = L_1 x_1 + \ldots L_n x_n$ is modeled linearly by the following (in)equalities (cf. [Kal02], p.116):

$$\begin{aligned} |g(x)| &= a^+ + a^- && a^+, a^- \in \mathbb{R}^+ \\ g(x) &= a^+ - a^- \\ a^+ &\leq M \cdot c && c \in \{0,1\}, M = \max|g(x)| \\ a^- &\leq M \cdot (1 - c) \,. \end{aligned}$$

Therefore, equation (4.18) can be replaced by the set of constraints

$$\begin{aligned} d_{rt} &= a_x^+ + a_x^- + a_y^+ + a_y^- \\ x_r - x_t &= a_x^+ - a_x^- \\ a_x^+ &\leq M \cdot c_{xrt} \\ a_x^- &\leq M \cdot (1 - c_{xrt}) \\ y_r - y_t &= a_y^+ - a_y^- \\ a_y^+ &\leq M \cdot c_{yrt} \\ a_y^- &\leq M \cdot (1 - c_{yrt}) \end{aligned}$$

with $c_{rt,x}, c_{rt,y} \in \{0,1\}, M_1 = \max|x_r - x_t|, M_2 = \max|y_r - y_t|$.
For every combination of robot r and target t two binary variables $c_{rt,x}$ and $c_{rt,y}$ are introduced,

which yields a total of $2n_R n_T$ additional binary variables compared to the former modeling approach.

While good results were obtained with an online controller, the increase of binary variables impeded the computation of an explicit controller. MATLAB's maximum variable size was already exceeded when solving a minimal problem setting.

4.5 Quadratic Problem Formulation

Representing the Target Observation Problem as a quadratic CFTOC is another attempt of reducing the problem complexity and shortening the duration of the explicit controller computation. In addition, quadratic cost functions are easier to tune and the solution is smoother around the reference point than in the linear case [Chr06].

The quadratic structure of the cost function

$$\min_{U_N} \quad \|Px(N)\|_2 + \sum_{k=0}^{N-1} \|Q_1 u(k)\|_2 + \|Q_2 \delta(k)\|_2 + \|Q_3 z(k)\|_2 + \|Q_4 x(k)\|_2 \tag{4.19}$$

offers the possibility to minimize the exact (squared) Euclidean robot-target distances of the form $(x_r - x_t)^2 + (y_r - y_t)^2$. Therefore, the variables d_{rt} can be eliminated in the MLD model (4.14b) - (4.14d). For this purpose, the model has to be adapted accordingly.

While the constraints describing motion dynamics (4.1) remain unchanged, the distance approximation (4.2) is no longer needed and can be excluded. A modification of observation constraint (4.3) is necessary due to the missing accessibility of the distances d_{rt}.
Robot r observes target t when t is located within r's observation range with radius R. This condition is approximated similar to (4.2) by the implication

$$b_{rt} = 1 \quad \Rightarrow \quad (x_r - x_t) sin\frac{2\pi j}{n_d} + (y_r - y_t) cos\frac{2\pi j}{n_d} \leq R \,. \tag{4.20}$$

Applying the Big-M technique, (4.20) transforms into

$$(x_r - x_t) sin\frac{2\pi j}{n_d} + (y_r - y_t) cos\frac{2\pi j}{n_d} \leq R + M(1 - b_{rt}) \,, \tag{4.21}$$

where $M \geq max((x_r - x_t) sin\frac{2\pi j}{n_d} + (y_r - y_t) cos\frac{2\pi j}{n_d} - R)$.
While the constraints (4.6) stay the same, the inequalities (4.9) are eliminated. This causes a slight change in the structure of the cost function (4.13). While the linear CFTOC (4.14) only minimizes distances to unobserved targets, the quadratic version will minimizes *all* robot-target distances regardless of a target's observation status. Therefore, (4.11) becomes

$$\sum_k \sum_{r,t} (x_r^{(k)} - x_t^{(k)})^2 + (y_r^{(k)} - y_t^{(k)})^2 \tag{4.22}$$

Since the matrices Q_i in a quadratic cost function have to be positive semi-definite, (4.22) cannot be described in direct dependence on the corresponding components of vector $x(k)$. Hence, additional auxiliary variables

$$ax_{rt} = x_r - x_t \quad \text{and} \quad ay_{rt} = y_r - y_t \tag{4.23}$$

have to be introduced.

Due to the additional inequalities representing (4.23), the number of constraints compared to the linear CFTOC remains unchanged. The quadratic problem formulation might still provide a benefit in computational performance compared to the linear performance index. Due to the different distance approaches, however, the quality of the solutions has to be considered, as well.

4.6 Stability and Optimality

In both the linear (see section 4.3) as well as the quadratic (see section 4.5) problem formulation the number of constraints were kept at a minimum in order to speed up the controller computation. As already mentioned, this strategy works at the expense of certain properties of the MLD model.

In both representations of the Target Observation Problem the matrices B_2, D_2, B_3, and D_3 of the MLD systems equal zero and, therefore, condition *(ii)* of definition 1 becomes redundant. This makes it difficult to directly transfer or apply definition 5 and theorem 1 to the MLD system modeled here. Changing the model to be well posed would make it much more complex and the problem more difficult to solve and, thus, would permit the multi-parametric computation of an explicit controller. However, one can assume that if the missing constraints and variables needed to obtain a well posed model were added to the present model version, the optimization would still provide nearly identical results. For this reason a complete characterization of the modeled MLD systems will not be considered. Nevertheless, a short descriptive discussion of stability and optimality for the Target Observation Problem will be given in the following.

In the problem description (TOP) an *equilibrium state* can be considered to be any combination of x, u, and δ variables, where all velocities and control inputs equal zero and a maximum number of targets is observed, i.e. as many components s_t of δ as possible for the considered robot/target constellation equal zero. Equilibrium states and hence also the optimal solution to the problem cannot be uniquely determined. This is due to the possible interchangeability of certain robots and targets on the one hand and due to the equivalence of different robot positions with respect to the maximum number of observed targets on the other.

One can introduce certain "equilibrium *areas*" in the neighborhood of the targets. If only a single target T is considered, its equilibrium area is the neighborhood with radius R and center (x_T, y_T). For more than one target, equilibrium areas can occur as the intersection of target neighborhoods if those are located close enough to each other. Figure 4.1 shows two examples of equilibrium areas for one target and two targets, respectively.

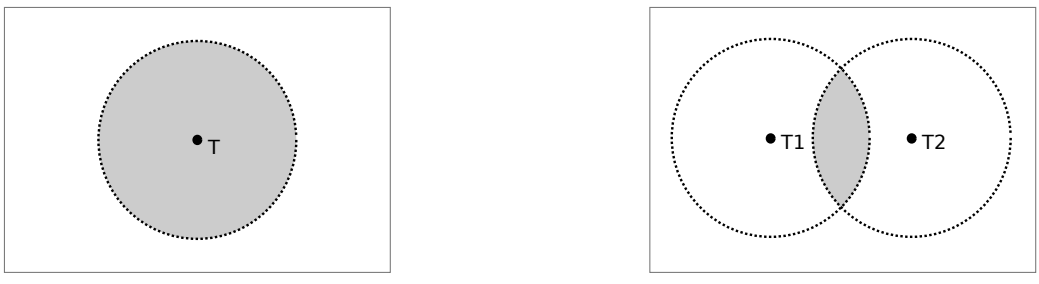


Figure 4.1: Examples of equilibrium areas for one and two targets, respectively.

As long as a robot r stays within an equilibrium area $\mathcal{S}$, the corresponding discrete structure of the system state, i.e. the values of $b_{rj}, j = 1 \dots n_T$, does not change. Considering a single robot, one can speak of local asymptotical stability (cf. definition 4) of robot positions $(x_r, y_r) \in \mathcal{S} \setminus \partial\mathcal{S}$. In case a disturbance moves the robot's position out of the area, the controller will drive it either back inside the same area or inside a different area closer to the robot's current position. The latter case only arises if the corresponding discrete state structure for a robot located in the new equilibrium area yields the same objective value as for the former area.

In this context, one can classify the equilibrium areas with respect to the corresponding objective value. The more target neigborhoods intersect, the more favourable is a robot position inside the area, see figure 4.2.

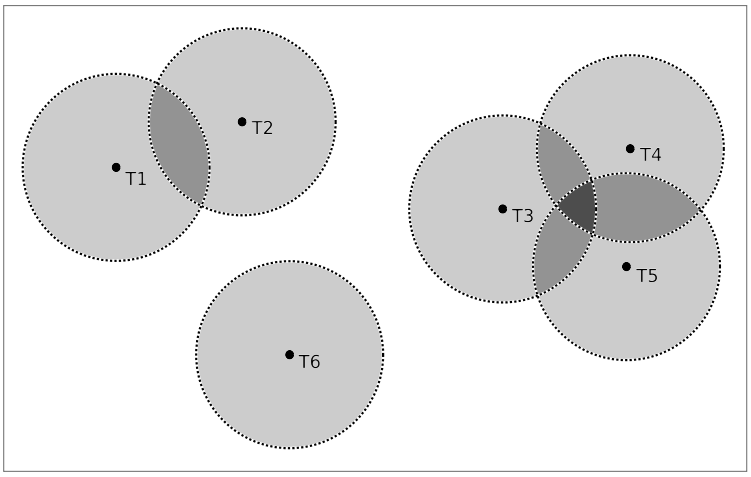


Figure 4.2: Different equilibrium areas as intersections of target neighborhoods: The darker the color, the more favourable are robot positions in the area.

In a scenario with multiple robots, the priority of the existing equilibrium areas can change as soon as a target is observed by one of the robots. From the view of all other robots the neighborhood of the observed target ceases and the controller drives them towards favourable positions with regard to the unobserved targets.

It decisively depends on the number of robots and targets, their initial positions, their maximum velocities, and in particular on the prediction horizon N whether or not an equilibrium is reached within the time horizon T and whether it represents a global optimum for the Target Observation Problem. Clearly, an increase of the prediction horizon N will improve the solution.

In summary, it can be stated, that the controller based on problem description (TOP) guarantees "local stability" of robot positions inside the equilibrium areas and "global stability" with respect to the optimal objective value.

5 Results

5.1 Explicit Controller Computation

Several attempts to compute an explicit model-predictive controller as described in section 3.3 were made within the scope of this thesis. It turned out, that the large number of binary variables in the problem description (TOP) leads to a prohibitive number of regions, which have to be generated when solving the corresponding multi-parametric program. For a minimal setting consisting of 2 robots, 2 targets, and prediction horizon $N = 3$, the mpMILP contains 18 binary variables. All attempts to reduce this number (cf. section 4.4) did not succeed. Solving this mpMILP would have required the solution of multiple mpLP with more than 100.000 regions each. Using the quadratic problem formulation presented in section 4.5, i.e. solving the corresponding mpMIQP, reduced the number of regions to around 50.000 per mpQP. However, on a 64Bit Linux system with DualCore CPU, 128 Gigabytes RAM, and a 64Bit Matlab version the computation of an explicit solution did not complete within 2 weeks and is assumed to last at least 3-4 weeks.

In addition, the CPLEX calls repeatedly performed during the computation required a permanent connection to the remote license server. Since the MPT does not provide functionality to catch license call errors or to buffer already obtained parts of the solution, all data would be lost as soon as any kind of error ocurred while solving the multi-parametric program.

The possibility to parallelize the explicit solution computation was also taken into account. However, the algorithms implemented in the MPT (cf. section 3.5) cannot easily be divided into independent iterations. A more detailed analysis of the MPT mechanisms and the underlying source code with respect to parallelization would have gone beyond the scope of this thesis. Therefore it was not possible to perform a parallel computation, e.g. on a cluster of multiple CPUs, apart from the fact that several independent Matlab jobs would each have required their own Matlab license.

Due to all the difficulties that arised during the attempts to compute an explicit model-predictive controller for the Target Observation Problem, more emphasis was put on further developping and evaluating the online controller.

5.2 Evaluating Quality and Efficiency of the MPC Solutions

In this section, a version of the Target Observation Problem with non-moving targets is considered. This problem can be solved in an open-loop fashion by determining the optimal solution of a MILP as described in section 4.3. A comparison of the resulting open-loop optimal control sequence and the closed-loop control sequences obtained from different model-predictive online controllers is presented. For this purpose, the performance of nine different online controllers for $N = 1, \ldots, 9$ will be investigated in terms of computational efficiency as well as the quality of the control sequence they provide.

Setup

The work area in the test scenario is considered a square of size 8000×8000 units. Initially, robots and targets are randomly located inside this work area, the robots' velocity v and acceleration u equals zero. During the simulation, the robots' velocity can vary between -200 and 500 units per second. The same bounds in units per square second apply to their acceleration. Each robot has an observation range of radius $R = 2500$.

Five sets of simulations were considered, one for each number of robots n_R between 1 and 5. The number of targets was considered $n_T = 2n_R$, which represents a problem version where, in general, no complete target coverage can be achieved. For each instantiation of n_R, 50 simulation runs of length $T = 20$ time steps were performed. A MILP was solved for each setting of initial robot and target positions. The same settings were then simulated using nine different model-predictive online controllers with prediction horizons varying from $N = 1$ to $N = 9$. Identical cost function weights (see section 4.2) were used for the overall MILP as well as the CFTOC problems (4.14) solved during the model-predictive control procedure:

$$\begin{aligned} q_z &= 0.5 \ , \\ q_\delta &= 10000 \ , \text{and} \\ q_u &= 10 \ . \end{aligned}$$

These weights mirror the objective priorities as the main objective is to minimize the number of unobserved targets. Hence q_δ is very large. q_z weights the minimization of the robots' distances to unobserved targets and is required to move them towards convenient positions in order to observe as many targets as possible. The weight on the control inputs q_u is to ensure control efficient robot trajectories. In the system models, the robots' observation range is approximated by an octagon, i.e. $n_d = 8$, fitting into a circle with radius 2500.

The obtained simulation runs were evaluated with respect to the required computational time, the achieved target coverage over the entire time horizon, and the number of observed targets at the final time step. The target coverage is measured by the "*A metric*" Parker introduced in [Par02]:

$$A = \frac{1}{n_T} \sum_{t=1}^{T} \sum_{j=1}^{n_T} \frac{a_j^{(t)}}{T} \ , \qquad \text{(A metric)}$$

where $a_j^{(t)}$ are binary variables that equal 1 if target j is observed at time step t and 0 otherwise. The *A metric* represents the average percentage of targets being observed by at least one robot at some instant of time throughout a simulation of T time units.

In order to compare the final target observation status, a separate value of A for the last time step only is measured by

$$A_{final} = \frac{1}{n_T} \sum_{j=1}^{n_T} a_j^{(T)} \ . \qquad (5.1)$$

All computations were performed on a machine with DualCore CPU (Intel Pentium 4, 3GHz) and 3GB RAM. CPLEX [ILO07] was used as the standard MILP solver.

Figure 5.1 presents a comparison of the average time required to compute the MILP solution on the one hand, and the average time required to perform the corresponding simulation run using the online controller with prediction horizon N on the other hand.

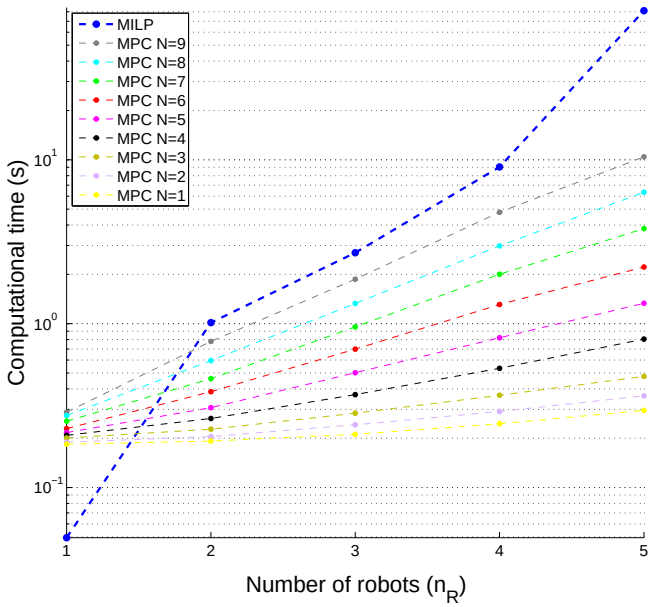


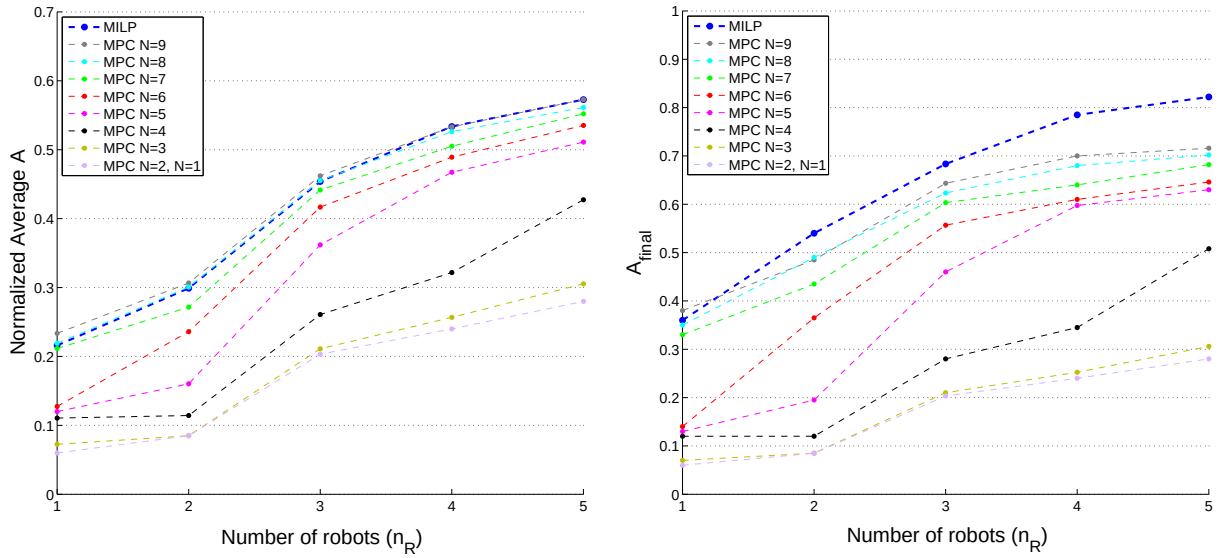
Figure 5.1: Average computational time required to solve the overall MILP for $n_R = 1,\ldots,5$ and $n_T = 2n_R$ compared to the average time required for one simulation run using a model-predictive controller with prediction horizon N.

The computational times increase exponentially with the number of vehicles involved in the problem, i.e. the number of constraints. Since in the model-predictive approach a MILP over the prediction horizon N has to be solved at every time step t, the computational effort grows the faster the greater the value of N. However, it is evident from figure 5.1, that solving T MILP over time horizon $N < \frac{T}{2}$ is more efficient than solving one MILP over the entire time horizon T, especially for large numbers of constraints.

This conclusion does not apply to the case $n_R = 1$, though. As can be seen here, solving the overall MILP is much faster than the online controller simulation. It can be stated, that for very low numbers of constraints, the time to solve the MILP is exceeded by the time, the online controller requires to process the obtained data.

Figure 5.2 shows a comparison of the *A metric* and the values of A_{final} obtained from the MILP solution and the different online controllers, respectively. Clearly, all curves rise when the number of robots (hence also the number of targets) increases. The denser the spatial distribution of the targets, the easier is it for the robots to reach them and the more likely is it that one robot can observe more than one target at the same time.

Regarding the graphs representing A_{final} in figure 5.2 (b), one can see how the MPC solution improves when N increases. However, even with the largest considered prediction horizon, $N = 9$, the online controller was not able to drive the robots to the optimal final positions as obtained from the MILP solution. This is because the controller can only plan optimal robot movements



(a) (b)

Figure 5.2: Comparison of (a) the *A metric* and (b) the target coverage at the final time step $t = T$, obtained from the optimal MILP solution and the model-predictive controllers (prediction horizon N), respectively.

within the prediction horizon N, i.e. at each time step a "local" optimum for the next N time steps is computed. The resulting sequence of T "local" optima not necessarily leads to the global optimum of the overall MILP. If robot r is not able to reach target t within the next N time steps, the online controller will drive it to a closer target. Hence, too distant targets are likely to stay unobserved. Due to the considered simulation setup, cases occur in which a target cannot even be reached within T time steps. This explains, why the blue MILP graph in figure 5.2 (b) does not start at a value of $A_{final} = 0.5$ for $n_R = 1$.

The graphs in figure 5.2 (a) representing the *A metric* are very similar to the ones in 5.2 (b). However, for N large enough, the A values obtained with the MPC approach come very close to or even go beyond the optimal values. Although the final robot positions are not optimal and some of the targets might never get observed, the robots controlled by the online controller stick with their "local" target for a longer period in time than the robots in the optimal solution. That is why the MPC approach in this case provides very good results since A records the average percentage of observed targets for *all* time steps t.

5.3 Comparison to an Existing Heuristic Approach

This section presents a comparison of the online MPC approach with the heuristic decentralized A-CMOMMT approach Parker proposed in [Par02] based on an observation scenario with multiple moving targets. An almost identical simulation setup as in [Par02] was used and will be described in the following.

Setup

The area under consideration is a square with no obstacles. Initially, robots and targets are randomly positioned within a 2000×2000 square in the center of the work area. Since the robots' observation range has a fixed radius of 2600 units, at the beginning all targets are observed. During the simulation, they move at a constant speed of 150 units per second and with a 5% chance at every time step to change their orientation between $-90°$ and $90°$. If a target reaches the boundary of the work area, it is reflected off the boundary. A total of $T = 120$ time steps of $\Delta t = 1$ second is considered. The robots can vary their speed between -200 and 200 units per second in both x- and y-direction. A maximum acceleration of $u_{max} = \pm 200$ is considered. Furthermore the following parameters are used:

$$\begin{aligned} N &= 5 \\ n_d &= 8 \\ q_z &= 0.5 \\ q_\delta &= 10000 \\ q_u &= 10 \ , \end{aligned}$$

where N is the prediction horizon, n_d represents the number of hyperplanes to approximate the observation range, and q_z, q_δ, q_u weight the cost function as described in section 4.2. q_δ and q_u are selected as described in the previous section. q_z controls the minimization of distances to unobserved targets and has to be tuned carefully. A bigger value of q_z would cause the robots to move towards the "center" of all unobserved targets, leaving the one they were already observing.

The evaluation of each simulation run is based on measuring the *A metric* [Par02]

$$A = \frac{1}{n_T} \sum_{t=1}^{T} \sum_{j=1}^{n_T} \frac{a_j^{(t)}}{T} \ , \qquad \text{(A metric)}$$

as introduced in the previous section, where n_T the number of targets, T the considered time horizon and $a_j^{(t)}$ binary variables, that equal 1 if target j is observed at time step t and 0 otherwise.

Two sets of experiments were performed. In the first set, an equal number of robots and targets is considered, i.e. $\frac{n_R}{n_T} = 1$, the second set considers a robot target ratio of $\frac{n_R}{n_T} = \frac{1}{4}$. For both sets, the number of robots is varied between 1 and 10, the number of targets accordingly between 1 and 20. Moreover, the size of the work area, defined by its side-length $2D$, varys between $D = 1000$ and $D = 50000$. For each instantiation of n_R, n_T, and D 250 simulation runs are performed and an average for A is derived. The normalization of the A metric with respect to the number of targets allows a comparison of experiments that vary in the quantity of targets.

Differences in the Simulation Setup

Although an identical setup of the experiments was aspired, a few differences between the A-CMOMMT approach and the here developed MPC approach, e.g. concerning the system models, have to be taken into account when evaluating the comparison. Table 5.1 gives an overview of the differences.

	A-CMOMMT appoach	MPC approach
Work Area	Circular work area with radius R between 1000 and 50000	Square work area with side-length $2D$ and D between 1000 and 50000
Observation Range	Circular observation range with radius 2600 and "predictive tracking range" of 3000	Approximation of a circular observation range as largest octagon fitting into a circle with radius 2600
Robot Dynamics	Motion dynamics consist of orientation and a fixed speed of 200 units per second	Dynamic model (4.1) with variable x- and y-velocities (-200 to 200 units per second) and variable x- and y-accelerations (-200 to 200 units per square second)
Target Dynamics	Randomly assigned fixed speeds between 0 and 150 units per second	Fixed speed of 150 units per second
Collision Avoidance	Robots repel each other	No collision avoidance

Table 5.1: Differences between the compared approaches

Nearly all of the presented differences are slightly to the disadvantage of the MPC approach compared to the A-CMOMMT approach. This has to be taken into account when evaluation the results of the comparison. A circular work area clearly is smaller than a square with corresponding side-length and the approximation of the robot's observation range by an octagon forces them to get slightly closer to a target before they actually observe it. The two differences concerning the dynamics additionally complicate the system behaviour. The simple dynamic model used for the MPC approach possibly makes the robots less agile, but is still more realistic than Parker's model. For example, as opposed to Parker's robots, the robots in the MPC approach cannot perform a 180° turn in only one time step, while maintaining constant speed. In addition, the targets in Parker's problem version possibly move more slowly than 150 units per second, which can make it easier for the robots to keep them in view. On the other hand, cases in which the targets are even too slow for the robots can be imagined. Then the robots' constant velocity of 200 units per second in the A-CMOMMT approach could even permit a continuous observation.

Results

Figure 5.3 shows the comparison of the results Parker obtained from her CMOMMT approaches and those obtained from the MPC approach. The figure on the left provides an overview of the performance of four different approaches Parker compared in [Par02]. *Fixed* refers to a version where all robots stay in randomly defined positions in the work area for all instants in time while they move around randomly in the *Random* approach. *Local* denotes Parker's approach that uses local force vectors to attract robots to nearby targets and repel them from other robots. As opposed to the *A-CMOMMT* approach, represented by the solid line, the force vectors in the *Local* approach are not weighted. For a detailed description of Parker's approaches see [Par02].
The colored lines in the figure on the right show the obtained MPC results for different numbers

of robots and targets, respectively. The black line represents the average of all obtained results for each instantiation of D.

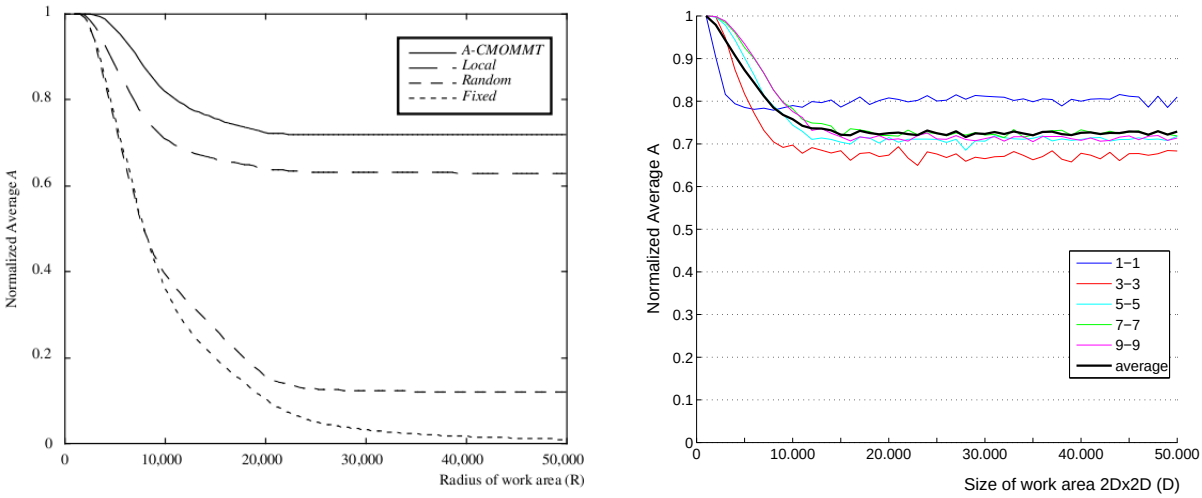


Figure 5.3: Quantitative comparison of the simulation results for a robot target ratio of $n_R/n_T = 1$ and randomly moving targets, obtained with (a) the A-CMOMMT approach [Par02] and (b) the MPC approach.

Both the A-CMOMMT and the MPC graph start at a 100% target observation for the smallest considered area. This is due to the size of the work area being smaller than the observation range of a single robot. The graphs then slope until they reach a more or less constant level. In the A-CMOMMT approach, a constant value of around 0.72 is reached, while the MPC graph varies slightly between 0.72 and 0.73. It can be noticed, that the constant level in the A-CMOMMT case is reached at a work area radius of around 20000 while the MPC graph slopes more quickly and reaches the constant level at around $D = 15000$. This difference corresponds to the 27% difference in the work area sizes. In both approaches a further increase of the work area size does not affect the value of the A metric since the targets are too slow to reach the boundary of the area. Hence, their reflection off the boundary "accidently" drives them into a robot's observation range.

For an equal number of robots and targets, a permanent 100% observation would be expected since each of the robots could be assigned a target. However, in the MPC approach targets are not assigned explicitly to each robot, but rather implicitly through the optimization problem. Figure 5.4 shows some examples of simulation results obtained for the robot target ration of 1 and $D = 10000$. Red markers and lines correspond to the targets, blue markers and lines to the robots enclosed in their observation range represented by the gray circles.

While (a) and (b) represent an ideal simulation progress with permanent target observation, the other figures show examples in which the (TOP) problem formulation cannot prevent targets from escaping the robots' observation. In (b), (c), (d), and (e) there is at least one non-moving robot that stays in the center of the area. This is due to the minimization of the control input u which corresponds to the robots' acceleration. Since at the beginning of the simulation all targets are covered by other robots, the remaining robot r does not have to move, i.e. $u_{r,x} = u_{r,y} = 0$. Later during the simulation when a target escapes observation, it depends on the prediction horizon

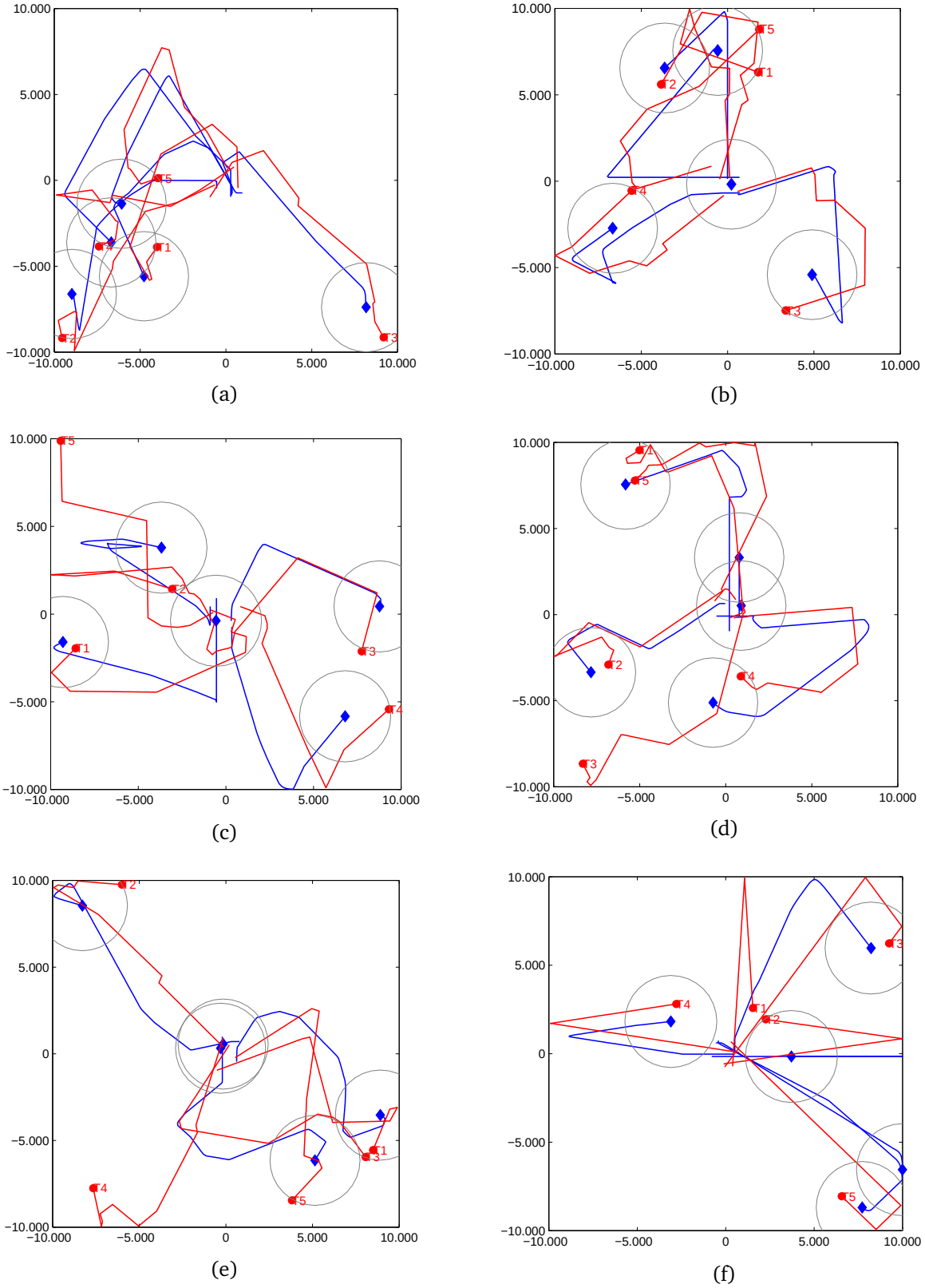
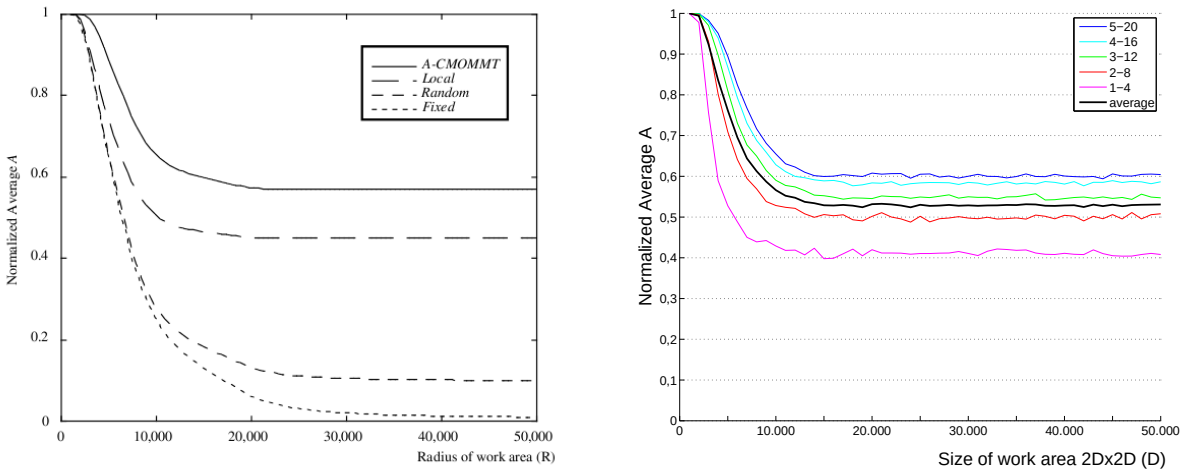


Figure 5.4: Simulation results for $n_R = 5$, $n_T = 5$, and $D = 10000$. Red markers represent targets, blue markers represent robots with their observation range (gray circle), the red/blue lines show the targets'/robots' movements in the x-y-plane over the time horizon $T = 120s$.

N whether robot r takes over or not. (c), (d), and (e) show simulations where the escaping target is too far away from the robot in the center. The robot cannot reach the target within the prediction horizon $N = 5$. Hence, it stays immobile due to the minimization of u. For the same reason, in figure (f) one of the robots moves towards the boundary in the lower right corner of the work area, although the nearby target is already observed by one of its teammates. The control input to stop the robot would "cost" more than to let it move on. In this point, the model of the Target Observation Problem lacks correspondence to reality. In the cases described above a greater prediction horizon could certainly improve the system behaviour.



(a) (b)

Figure 5.5: Quantitative comparison of the simulation results for a robot target ratio of $n_R/n_T = 1/4$ and randomly moving targets, obtained with (a) the A-CMOMMT approach [Par02] and (b) the MPC approach.

In figure 5.5 the quantitative results for a robot target ratio of 1/4 are shown. The curve shapes are similar to those in figure 5.3 though even out at a lower level clearly due to the fact that the targets outnumber the robots.

Figure 5.6 presents examples of typical simulation runs. The simulation depicted in (d) clearly results in a greater value for A than for the other cases. When the targets swarm out in a star-like manner as in (b) and (c), the robots move in those directions where more than one target can be observed by each robot, but still there is no chance to observe all of them.

Situations, in which one of the robots stays immobile as discussed for $n_R/n_T = 1$ are unlikely to occur for the ratio of 1/4 since there exist enough unobserved targets for all of the robots at the beginning of the simulation. Moreover, in the majority of the cases, a larger prediction horizon would not improve the robots' cooperative behaviour. Their future motions and the effect of which predicted by the system model are based on the target positions at the *current* instant in time. The model does not allow for somehow predicted target movements that would permit to plan the robots' behaviour more precisely. Hence, the obtained results are close the best that can be achieved for the considered target observation scenario. Their slight inferiority to Parker's results can be justified by the differences in the simulation setup discussed previously.

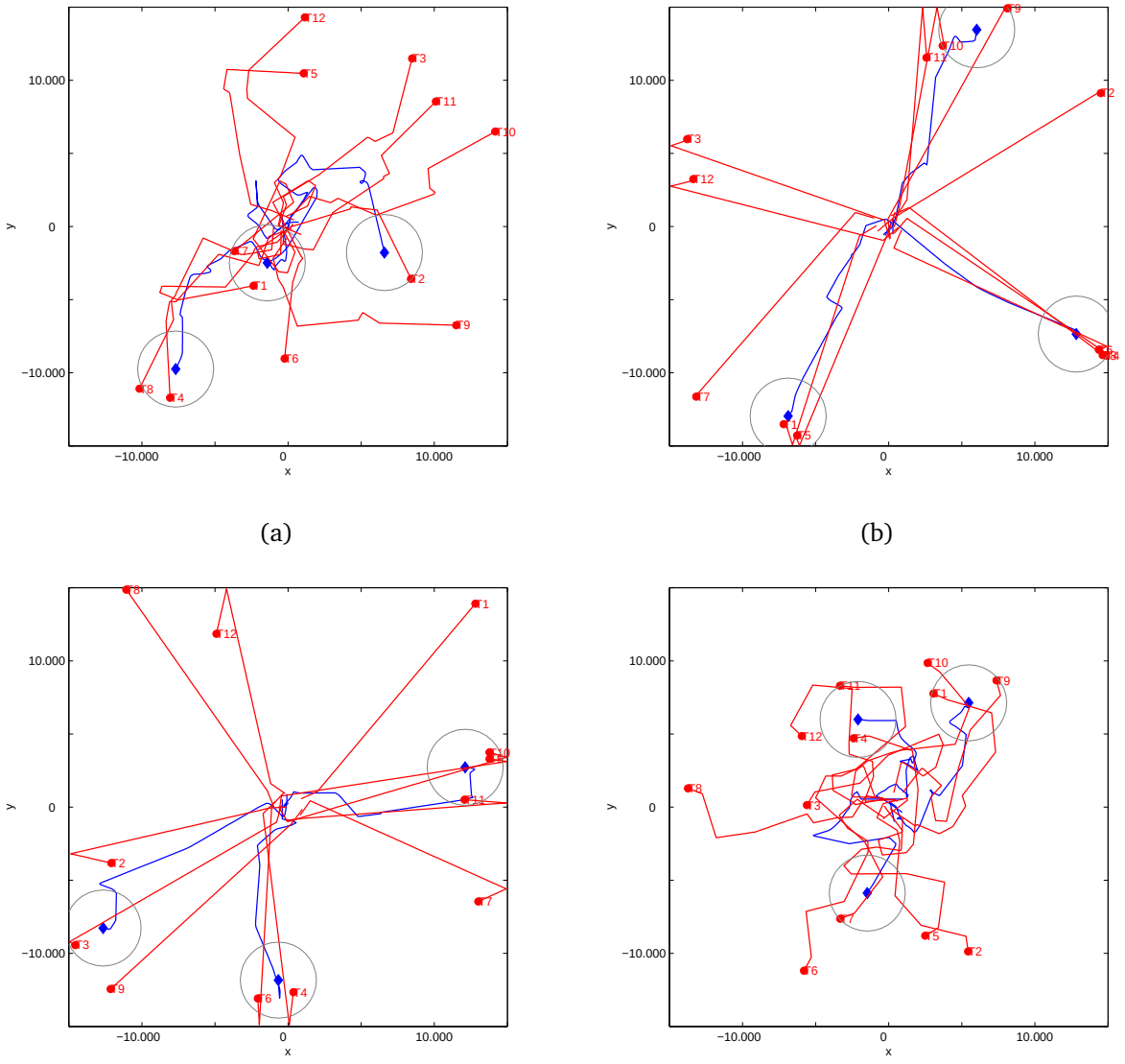


Figure 5.6: Simulation results for $n_R = 3$, $n_T = 12$, and $D = 15000$.

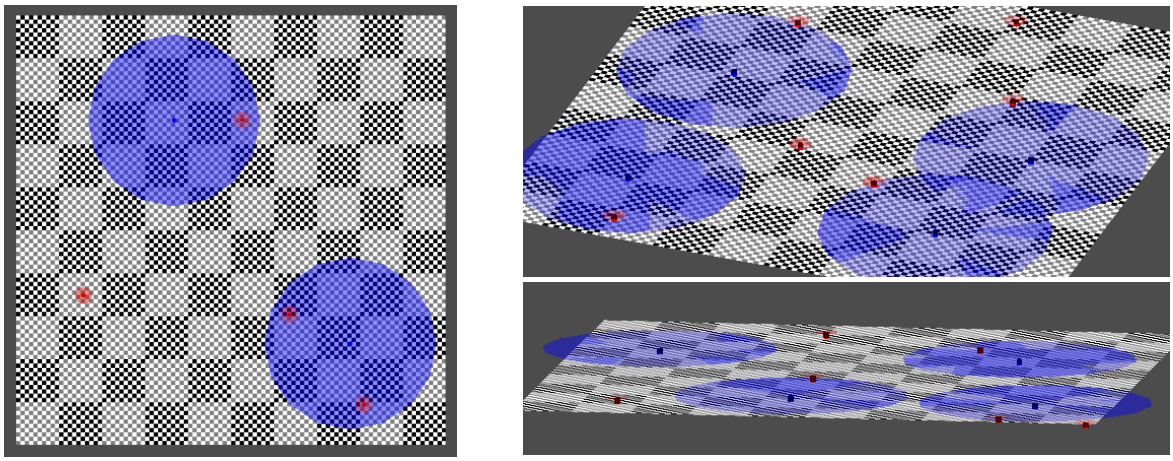
6 Discussion and Outlook

6.1 Application to Real Systems

Validation by Simulation

The discrete-time linear robot dynamics as introduced in section 4.1 can only approximate the physical behaviour of a real system. In addition, the considered model represents an idealized system without any disturbances. For these reasons, the resulting model-based controller has to be validated in applications to real systems.

A first step has been taken by providing an interface between the Multi-Parametric Toolbox for Matlab and the *Multi-Robot-Simulation-Framework* (MuRoSimF) developed at the Simulation, Systems Optimization and Robotics group (SIM) at TU Darmstadt. Based on a TCP/IP connection, which enables a frequent data exchange between the MPT and MuRoSimF, simulations of different target observation scenarios could be performed, see figure 6.1 for simulation snapshots.



(a) (b)

Figure 6.1: (a) Top view and (b) 3D views of a simulation performed with MuRoSimF. The work area is a 100×100 square. Robots are represented by blue cylinders surrounded by a blue transparent circle representing their observation range. Targets are depicted by red cylinders.

MuRoSimF offers the possibility to interactively test the controller in real time. The user can, for example, simulate disturbances on the system by dragging robots or targets away from their current positions. A detailed description of the simulation framework MuRoSimF and its features can e.g. be found in [FPvS08].

The online controller performed well in real-time simulations of smaller systems. For a problem involving 2 robots and 2 targets, it provided the next control input within $\sim$ 0.04-0.09 seconds (Intel Pentium 4 CPU, 2.66GHz, 1 GB RAM). This allowed a fast data exchange between controller

and simulation, keeping deviations from the desired system behaviour at a minimum. However, simulations involving more vehicles gave a first impression on how a communication delay can affect the system behaviour and cause gaps between current and desired values.

Decentralization

In order to overcome the dependence on a stable communication between the central controller and all robots, a decentralization of the model-predictive approach could be considered. Each robot could be controlled by its individual model-predictive controller. Based on a system model and the positions of team-mates and nearby targets currently perceived by the robot, the controller could provide its next optimal control input. Adaptations might be necessary if the robot detects more (or less) robots or targets than the system model includes. Nevertheless, the individual controllers in this approach would only have to cover parts of the complete system, which reduces each controller's complexity. Moreover, robots and targets could be added to or removed from the considered system without impairing the optimal control.

6.2 Conclusions

In this thesis, a model-predictive approach to control cooperative multi-vehicle systems was investigated by means of a benchmark scenario from target observation. The considered scenario was modeled as Mixed Logical Dynamical (MLD) System. Based on the MLD system, model-predictive online controllers were computed and tested.

The complexity of the cooperative control problem prohibited the computation of an explicit controller. Real-time control based on the explicit solution would be more efficient than the online controller. Moreover, the analysis of an explicit controller could have given further insight into the characteristics of the controlled system. In order to speed up the computation and to obtain an explicit solution, possibilities to parallelize the applied algorithms could be further investigated.

For problems with stationary targets, the efficiency and performance of the online controller was analysed based on a comparison to the global optimum of a corresponding MILP over the entire time horizon. For problems involving multiple moving targets, the online controller performed well compared to a decentralized heuristic approach proposed in [Par02].

An interface for the simulation framework MuRoSimF was provided, which offers the possibility to interactively develop, test, and evaluate model-predictive controllers designed with the Multi-Parametric Toolbox in Matlab. The performed simulations verified the real-time applicability of the online model-predictive approach, at least to small-scale multi-robot systems. Further tests would be needed to generalize this statement and to fathom benefits and limits of the online controller with respect to real-time applications. One possibility for further developing and improving the efficiency of real-time model-predictive control was presented in the previous section.

The presented investigations motivate further analysis of the model-predictive control scheme with respect to characteristics like robustness and stability. Its potential with respect to control of cooperative multi-vehicle systems was demonstrated and verified. If further improved, the model-predictive strategy can be expected to efficiently provide reliable optimal control of cooperative mobility in real-time applications.

Bibliography

[Bao05] M. Baotić. *Optimal Control of Piecewise Affine Systems - a Multi-parametric Approach*. PhD thesis, Swiss Federal Institute of Technology Zurich, 2005.

[Bar09] S. Bartsch. Formationswechsel bei Mehrfahrzeugsystemen: Modellierung und Lösung auf Basis von MILP (bachelor thesis; ongoing work). Master's thesis, Technische Universität Darmstadt, 2009.

[BBBM03] F Borrelli, M. Baotić, A. Bemporad, and M. Morari. An efficient algorithm for computing the state feedback optimal control law for discrete time hybrid systems. *Proceedings of the 2003 American Control Conference*, 6:4717–4722, 2003.

[BBM00] A. Bemporad, F. Borrelli, and M. Morari. Optimal controllers for hybrid systems: stability and piecewise linear explicit form. In *Proc. 39th IEEE Conference on Decision and Control*, volume 2, pages 1810–1815 vol.2, 2000.

[BBM02] A. Bemporad, F. Borrelli, and M. Morari. Model predictive control based on linear programming - the explicit solution. 47(12):1974–1985, 2002.

[Bem04] A. Bemporad. *Hybrid Toolbox - User's Guide*, 2004. `http://www.dii.unisi.it/hybrid/toolbox`.

[BF06] C. Branca and R. Fierro. A hierarchical optimization algorithm for cooperative vehicle networks. In *Proc. American Control Conference*, 2006.

[BFTM00] A. Bemporad, G. Ferrari-Trecate, and M. Morari. Observability and controllability of piecewise affine and hybrid systems. 45(10):1864–1876, Oct. 2000.

[BM99] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35:407–427, 1999.

[BMDP02] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38:3–20, 2002.

[Bor03] F. Borrelli. *Optimal Control of Linear and Hybrid Systems*. Springer, 2003.

[CDS+03] M. Campbell, R. D'Andrea, D. Schneider, A. Chaudhry, S. Waydo, J. Sullivan, J. Veverka, and A. Klochko. Roboflag games using systems based, hierarchical control. In *Proceedings of the American Control Conference*, 2003.

[Chr06] F. J. Christophersen. *Optimal Control and Analysis for Constrained Piecewise Affine Systems*. PhD thesis, Swiss Federal Institute of Technology (ETH) Zurich, 2006.

[CS05] G.C. Chasparis and J.S. Shamma. Linear-programming-based multi-vehicle path planning with adversaries. In *Proc. American Control Conference the 2005*, pages 1072–1077 vol. 2, 2005.

[DP00] V. Dua1 and E. N. Pistikopoulos. An algorithm for the solution of multiparametric mixed integer linear programming problems. *Annals of Operations Research*, 99:123–139, 2000.

[Dun07] W. B. Dunbar. *Cooperative Control of Distributed Multi-Agent Systems*, chapter Distributed Receding Horizon Control of Cost Coupled Systems. John Wiley & Sons, Ltd., 2007.

[ED05] Matthew G. Earl and Raffaello D'Andrea. Multi-vehicle cooperative control using mixed integer linear programming. *CoRR*, abs/cs/0501092, 2005.

[FPvS08] M. Friedmann, K. Petersen, and O. von Stryk. Simulation of multi-robot teams with flexible level of detail. In *Proceedings of the 2008 International Conference on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR)*, 2008.

[ILO07] *ILOG CPLEX 11.0 User's Manual*, September 2007.

[Kal02] J. Kallrath. *Gemischt-ganzzahlige Optimierung: Modellierung in der Praxis*. Vieweg, 2002.

[KBB07] T. Keviczky, F. Borrelli, and G. J. Balas. *Cooperative Control of Distributed Multiagent Systems*, chapter Distributed Predictive Control: Stability and Feasibility Issues. John Wiley & Sons, Ltd., 2007.

[KGBC06] M. Kvasnica, P. Grieder, M. Baotić, and F. J. Christophersen. *Multi-Parametric Toolbox (MPT)*. Swiss Federal Institute of Technology (ETH) Zurich, December 2006.

[Kva08] M. Kvasnica. *Efficient Software Tools for Control and Analysis of Hybrid Systems*. PhD thesis, Swiss Federal Institue of Technology (ETH) Zurich, 2008.

[KZ08] K. Kanjanawanishkul and A. Zell. A model-predictive approach to formation control of omnidirectional mobile robots. In *Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2008)*, Nice, France, September 22-26 2008.

[LSPB05] S. Luke, K. Sullivan, L. Panait, and G. Balan. Tunably decentralized algorithms for cooperative target observation. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, 2005.

[Mur07] R. M. Murray. Recent research in cooperative control of multi-vehicle systems. *International Conference on Advances in Control and Optimization of Dynamical Systems*, 129:571–583, 2007.

[NSH04] R. R. Negenborn, B. De Schutter, and J. Hellendoorn. Multi-agent model predictive control: A survey. Technical report, Delft Center for Systems and Control, Delft University of Technology, 2004.

[Par98] L. E. Parker. Alliance: An architecture for fault tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14 (2), 1998.

[Par02] L. E. Parker. Distributed algorithms for multi-robot observation of multiple moving targets. *Autonomous Robots*, 12:231–255, 2002.

[Par08] L. E. Parker. Distributed intelligence: Overview of the field and its application in multi-robot systems. *Journal of Physical Agents*, 2(2):5–14, March 2008.

[PL05] L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11:387–434, 2005.

[PT00] L. E. Parker and C. Touzet. *Distributed Autonomous Robotic Systems 4*, chapter Multi-Robot Learning in a Cooperative Observation Task, pages 391–401. Springer, 2000.

[RvS07a] C. Reinl and O. von Stryk. Optimal control of cooperative multi-robot systems using mixed-integer linear programming. In *RoboMat 2007 - Workshop on Robotics and Mathematics*, 2007.

[RvS07b] C. Reinl and O. von Stryk. Optimal control of multi-vehicle-systems under communication constraints using mixed-integer linear programming. In *ROBOCOMM 2007 - First International Conference on Robot Communication and Coordination*, 2007.

[TJB03] P. Tøndel, T. A. Johansen, and A. Bemporad. Evaluation of piecewise affine control via binary search tree. *Automatica*, 39(5):945–950, May 2003.