INTERNATIONAL JOURNAL OF INTELLIGENT CONTROL SYSTEMS
VOL. 11, NO. 3, SEPTEMBER 2006. 141-153

Hierarchical Optimization Strategies for Deployment of Mobile Robots

Carlo Branca and Rafael Fierro

***Abstract*— In this paper, we integrate model predictive control (MPC) and mixed integer linear programming (MILP) into a hierarchical framework suitable for solving optimization problems involving robotic networks. A critical issue in MPC/MILP applications is that the underlying optimization problem must be solved on-line. This creates a time constraint which is hard to meet when the number of robots and the number of obstacles increase. To alleviate this difficulty, we develop strategies that significantly improve the efficiency of a hierarchical, decentralized optimization scheme. As an application is considered a case of target assignment problem in urban-like environments. Numerical simulations verify the scalability of the algorithm to the number of robots and complexity of the environment.**

***Index Terms*— Hierarchical optimization, multirobot coordination, model predictive control, mixed integer linear programming.**

I. INTRODUCTION

Recent advances in communication, computation, and embedded technologies add support to a growing interest in developing cooperative robotic networks. The motivation for developing these systems stems from the recognition that many tasks can be performed more efficiently and robustly by a group of mobile agents rather than by a single robot acting alone. Additionally, the increased computing power and resource availability of a group of agents allows users to extend the range of robotics applications to tasks such as multi-point surveillance, distributed localization and mapping, and cooperative transportation.

In this work, we combine model predictive control (MPC) and mixed integer linear programming (MILP) in a constrained optimization algorithm for robot swarms. The key strategy is based on two intuitive steps: (i) replacing large problems with smaller problems which are computationally tractable; (ii) finding heuristics that take advantage of the structure of the problem and drastically reduce the number of constraints.

Generally, MPC algorithms rely on an optimization of a predicted model response with respect to the plant input to determine the best input changes for a given state. Either hard constraints (that cannot be violated) or soft constraints (that can be violated but with some penalty) can be incorporated into the optimization, giving MPC a potential advantage over passive state feedback control laws. However, there are possible disadvantages to MPC. In its traditional use for process control, the primary disadvantage is often considered to be the need for a good model of the plant. In robotics applications, the foremost disadvantage may be the computational cost, which is often negligible for slow-moving systems in the process industry.

When dealing with motion coordination problems of multiple robots, it might be required to model logic constraints, timing and non-timing constraints [24]. MILP allows the encoding of logical rules, decisions, and constraints into the optimization problem [5]. Additionally, MILP has the capability of handling nonconvex constraints making possible to model requirements such as obstacle avoidance or inter-robot collision avoidance. One of the major drawbacks of using MILP in a motion coordination algorithm is that obtaining a solution is NP-hard. Therefore, the computational time grows exponentially with the number of binary variables involved in the problem. To overcome this limitation, at least, two approaches are possible: (i) improving the algorithm used by the solver [36], [15]; (ii) using some techniques to reduce the number of binary variables present in the model [33], [10].

In this paper, we follow the second philosophy. Specifically, the motion coordination problem is divided into two levels. The higher-level takes care of the network global objective. The lower-level, on the other hand, provides services related to individual robot goals such as obstacle and inter-robot collision avoidance. Moreover, the lower-level is decentralized in the sense that each robot solves its private optimization problem. Also, to improve the scalability of the hierarchical optimization algorithm some heuristics are included in the problem formulation.

The remainder of the paper is organized as follows. In Section II, we review relevant work on optimization approaches for cooperative robots. Section III gives some background and mathematical preliminaries. We formulate the target assignment optimization problem in Section IV. Section V and Section VI describe a centralized algorithm and a hierarchical/decentralized algorithm to solve the optimization problem, respectively. The algorithms were tested on several scenarios including an urban environment. Section VII contains results and a comparison between the two algorithms. Finally, we draw conclusions in Section VIII.

II. RELATED WORK

Biology, computer, math and control communities have focused attention on how crowds of people, flocks of birds, or schools of fish move together in a decentralized, coordinated manner. Flocking is a collective behavior that emerges when

This work is supported in part by NSF grants #0311460 and CAREER #0348637, and by the U.S. Army Research Office under grant DAAD19-03-1-0142 (through the University of Oklahoma)

C. Branca and R. Fierro are with the MARHES Lab, School of Electrical & Computer Engineering, Oklahoma State University, Stillwater, OK 74078-5032, USA (e-mail: {carlo.branca, rfierro}@okstate.edu).

a large number of mobile agents having a common goal [31] interact with each other to reach a consensus state in the heading angles and intervehicle separations [30]. In [39], a simple model is proposed where several agents move with the same speed. Each agent aligns its heading based on the average of its own heading at time t and the headings of its surrounding neighbors. Despite the absence of a centralized coordination algorithm, the group is shown to align itself in the same direction and a flocking behavior emerges. The authors in [21] present a theoretical explanation for the observed behavior.

In recent years, robotic networks have been designed for various applications such as formation control [12], target assignment, task allocation [27], and swarm aggregation [19]. Moreover, a variety of optimization techniques [9], [8] are increasingly being applied to coordination of robotic networks engaged in distributed sensing tasks [11]. Optimal motion planning is considered in [4], [26] for multiple robots, and in [6] for marine vehicles. Distributed motion planning approaches are discussed in [25]. In [28] the authors study optimal sensor placement for mobile sensor networks. The task of repositioning a formation of robots to a new shape as defined in [13] while minimizing either the maximum distance that any robot travels, or the total distance traveled by the formation is described in [38].

Decentralized and distributed MPC algorithms are being explored in [14], [35], [23], [18]. Furthermore, several authors are using MILP in applications ranging from ground robots to unmanned aerial vehicles [34], [20], [2]. Since, the computation time is a major concern in MILP and MPC formulations, several researchers are developing algorithms that aim to improve the efficiency and scalability of the solution methods [37], [10], [16]. Finally, a distributed optimization approach is developed and applied to formation fight in [32].

III. Mathematical Preliminaries

A. Model Predictive Control

Model predictive control (MPC) or receding horizon control (RHC) is a control algorithm in which the control input is obtained by solving, at each sampling time, an on-line finite horizon open-loop optimal control problem, using the current state of the system as initial state. The solution of the optimization problem provides a sequence of control inputs but only the first element of the sequence is applied to the system [29]. Generally, control problems have to consider constraints such as actuator saturation and safety limits. Model predictive control has become a valuable tool to solve constrained control problems.

Consider a discrete-time system given by

$$\begin{aligned} x(k+1) &= f(x(k), u(k)), \\ y(k) &= h(x(k)), \end{aligned} \tag{1}$$

where the state x and the input u must satisfy $x(k) \in \mathbb{X} \subset \mathbb{R}^n$ and $u(k) \in \mathbb{U} \subset \mathbb{R}^m$. The control objective is to drive in optimal way the state of the system to the origin while satisfying state and input constraints.

A cost function that measures the optimality of the solution is

$$J(x, \mathbf{u}, k) = \sum_{i=k}^{k+T-1} c(x(i), u(i)) + C(x(k+T)), \tag{2}$$

where $\mathbf{u} = \{u(k), u(k+1), \ldots, u(k+T-1)\}$, T is called the receding or control horizon, c is the cost that penalizes the trajectory and the input, and C is the final cost associated with the final state of the system. Under the assumption that $f(\cdot)$ and $c(\cdot)$ are time-invariant, the cost function is also time-invariant. In other words, the solution of the optimization problem with the system in state x at time k is the same as the solution of the optimization problem from state x at time 0 ($\mathbf{u}^o(x,k) = \mathbf{u}^o(x,0)$).

The open-loop optimization problem can be defined as

$$\mathcal{P}_T(x) : J_T^o(x) = \min_{\mathbf{u}} \{J_T(x, \mathbf{u}) | \mathbf{u} \in \mathcal{U}\}, \tag{3}$$

where, now,

$$J(x) = \sum_{i=0}^{T-1} c(x(i), u(i)) + C(x(T)) \tag{4}$$

and $\mathcal{U}$ is the set of admissible control inputs that satisfies all the constraints. The MPC law is given by

$$\kappa_T(x) := u^o(0, x). \tag{5}$$

Early versions of MPC algorithms did not guarantee stability. Therefore, several modifications have been reported in the literature to achieve this goal. In [7], the authors show that the original MPC model guarantees stability for unconstrained linear systems and constrained stable systems. A modification that ensured stability for time-varying, nonlinear, constrained discrete-time systems is reported in [22]. Specifically a terminal equality constraint $x(T) = 0$ is added to the problem $\mathcal{P}_T(x)$. This modification ensures stability but it brings up a feasibility problem. In fact, it is possible that a feasible solution cannot be found if the horizon T is not long enough. Determining the region of attraction of the controller (*i.e.*, a set $X_T \subset \mathbb{R}^n$ from which a feasible solution exists) becomes an important issue.

B. Mixed Integer Linear Programming

There is a wide range of practical problems that can be modeled with discrete and continuous variables, and linear constraints. Discrete optimization shows all its power and flexibility when decision variables are used. Decision variables are discrete variables that can be either zero or one (true or false). For example, if a new plant needs to be built in one among several possible locations, a discrete optimization problem can be modeled using decision variables. In this case there is a decision variable δ_i for every location that is one if the new plant is assigned to location i or zero otherwise.

Other discrete variables are indicator variables. These variables as the decision variables, can take only the value zero or one. However, they usually indicate the state of certain continuous variables. For example, suppose it is

necessary to know if $f(x) \leq 0$, an indicator variable δ can be used such that $f(x) \leq 0$ implies $\delta = 1$.

Given a set of logical statements involving decision and indicator variables, we would like to convert it into a set of mixed integer linear constraints. Decision variables are always associated with a statement. For example "$f(x) \leq 0$" or "the i-th task is scheduled in the j-th machine". It is common practice to represent statements with literals X_i that have a *truth value* of either "T" (True) or "F" (False). Linear equations involving decision variables can be used to obtain logical relations among different statements. For example, if the decision variable δ_1 is associated with statement X_1, and δ_2 to X_2, then adding the constraint

$$\delta_1 + \delta_2 \geq 0, \tag{6}$$

is equivalent of having a logical OR between the statement X_1 and X_2. A list of conversions of the most common logical operators is provided below

$$\begin{array}{lll} X_1 \text{ AND } X_2 & \text{is equivalent to} & \delta_1 = 1,\ \delta_2 = 1, \\ \text{NOT}(X_1) & \text{is equivalent to} & \delta_1 = 0, \\ X_1 \rightarrow X_2 & \text{is equivalent to} & \delta_1 - \delta_2 \leq 0, \\ X_1 \leftrightarrow X_2 & \text{is equivalent to} & \delta_1 - \delta_2 = 0, \\ X_1 \oplus X_2 & \text{is equivalent to} & \delta_1 + \delta_2 = 1. \end{array} \tag{7}$$

These logical operators can be combined to model complex logical statements.

An important tool in mixed integer linear programming is the **big M** technique [40]. This technique allows mapping of indicator variables into continuous variables. The basic building block for this technique is the translation of the implication

$$x > 0 \quad \rightarrow \quad \delta = 1, \tag{8}$$

that means $x > 0$ implies $\delta = 1$. The implication in (8) can be converted into the following mixed integer linear constraint

$$x - M\delta \leq 0, \tag{9}$$

where M is a large positive number. To verify the validity of the constraint in (9), it is useful to rewrite the inequality in the following way

$$x \leq M\delta. \tag{10}$$

In this second form it is easy to see that if $x > 0$, the only way to satisfy the constraint is by having $\delta = 1$. Note that, if $x \leq 0$ the constraint is satisfied for every δ.

Let us consider the following implication

$$\delta = 1 \quad \rightarrow \quad x > 0. \tag{11}$$

The implication in (11) must be slightly modified to be converted into a mixed integer linear constraint. The modified implication becomes

$$\delta = 1 \quad \rightarrow \quad x \geq \epsilon, \tag{12}$$

where ϵ is the smallest number for which x is considered to be not zero (ϵ is usually the machine precision). The implication in (12) is translated in the following constraint

$$x - \epsilon\delta \geq 0. \tag{13}$$

It can be easily seen that if $\delta = 1$ the constraint in (13) forces x to be greater than or equal to ϵ, while if $\delta = 0$ no constraint is forced on x. We assume that the linear problem is in standard form; therefore, all continuous variables are greater than or equal to zero.

It is useful to know how to translate the following type of implications

$$\delta = 1 \quad \rightarrow \quad \sum_j a_j x_j \leq b. \tag{14}$$

This implication can be converted into the following linear constraint

$$\sum_j a_j x_j \leq b + M(1 - \delta), \tag{15}$$

where, as before, M is an upper bound on the expression $\sum_j a_j x_j - b$.

Finally, consider the following nonlinear expression

$$y = \delta f(x), \tag{16}$$

where δ is a binary variable and $f(x)$ is a linear function. The expression is converted into linear constraints by

$$\begin{array}{ll} y \leq M\delta, & y \geq m\delta, \\ y \leq f(x) - m(1 - \delta), & y \geq f(x) - M(1 - \delta), \end{array} \tag{17}$$

where M and m are respectively the upper and lower bounds on the function $f(x)$. It can be verified that if $\delta = 0$ the first two equations force y to be zero, while the second two are satisfied for every value of y. If instead $\delta = 1$, the first two equations are satisfied for every value of y, while the combination of the second two forces y to be equal to $f(x)$.

IV. Problem Statement

We consider a network of N_R mobile robotic sensors, each modeled as a point mass with damping factor, that is

$$\begin{aligned} \ddot{x}_i &= -b_i \dot{x}_i + u_{xi}, \\ \ddot{y}_i &= -b_i \dot{y}_i + u_{yi}, \end{aligned} \tag{18}$$

where x_i, y_i, u_{xi}, u_{yi}, and b_i are the x-position, y-position, acceleration on the x and y directions, and damping factor of robot i, respectively. We assume $(u_{xi},\ u_{yi}) \in U$ and $(\dot{x}_i,\ \dot{y}_i) \in V$ are bounded convex subsets of $\mathbb{R}^2$.

Since in this paper MPC is used in discrete-time, the discretized version of (18) becomes

$$\begin{pmatrix} x(k+1) \\ v_x(k+1) \\ y(k+1) \\ v_y(k+1) \end{pmatrix} = \begin{pmatrix} 1 & \Delta T & 0 & 0 \\ 0 & 1 - \Delta T\, b & 0 & 0 \\ 0 & 0 & 1 & \Delta T \\ 0 & 0 & 0 & 1 - \Delta T\, b \end{pmatrix} \begin{pmatrix} x(k) \\ v_x(k) \\ y(k) \\ v_y(k) \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ \Delta T & 0 \\ 0 & 0 \\ 0 & \Delta T \end{pmatrix} \begin{pmatrix} u_x(k) \\ u_y(k) \end{pmatrix},$$

where $v_x = \dot{x}$ and $v_y = \dot{y}$, and ΔT is the sampling time of

the system. In a more compact way, we have

$$X(k+1) = A\,X(k) + B\,U(k), \tag{19}$$

with the obvious meaning of the symbols. We assume robots are able to communicate with a base station, and sense position and velocity of other members of the team within their sensing range.

Let $\mathcal{T} = \{t_1, \ldots, t_{N_T}\}$ be a set of targets. Each target is defined by coordinates $t_i = (x_i^t,\ y_i^t)$. Also, we consider that there are N_o prohibited zones in the environment where the presence of robots is precluded. These areas are represented by convex sets $\mathbf{O}_i \subset \mathbb{R}^2$. Now we state the coordination problem considered in this work.

Problem 4.1: Given a network of N_R robotic sensor determine an optimal control algorithm that drives the network to visit all the targets in $\mathcal{T}$, while avoiding inter-vehicle collisions and prohibited zones $\mathbf{O} = \bigcup_{i=1}^{N_o} \mathbf{O}_i$.

A. Obstacles

The robots move in an environment populated with obstacles or unsafe zones described by convex linear sets that can be represented by linear inequalities of the form

$$OX \leq r, \qquad X = \begin{pmatrix} x \\ y \end{pmatrix}, \tag{20}$$

where O is an $R \times 2$ matrix, R is the number of linear constraints needed to define the obstacle, and $r \in \mathbb{R}^R$. Although, the use of this obstacle representation may seem restrictive, it is possible to describe a wide range of situations. For example, linear nonconvex obstacles can be modeled by composing convex sets. Similarly, a nonlinear obstacle can be approximated to a linear one as shown in Fig. 1.

Since a point mass robot is assumed, obstacles must be enlarged to consider the actual dimensions of the robots. Moreover, obstacles must also be enlarged due to discretization. Although, the MPC algorithm guarantees obstacle avoidance at every sampling time, it would be possible to have a trajectory that would end up in a collision in continuous time. This is most likely to happen in the vicinity of a corner of an obstacle as depicted in Fig. 1(a). The enlargement depends on the maximum possible distance D_{pp} between two points a robot can travel between samplings.

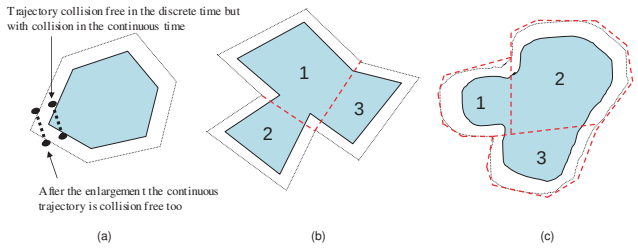


Fig. 1: Obstacle representations: (a) A convex linear obstacle enlarged to consider the robot's dimensions, (b) a nonconvex linear obstacle obtained by composition of convex linear sets, and (c) a nonconvex nonlinear obstacle represented using linear approximation and composition of convex linear sets.

D_{pp} is function of the maximum velocity of the robots $V_{\max}$, and of the sampling period ΔT, that is $D_{pp} = V_{\max}\Delta T$. Once an obstacle is enlarged, the following implication needs to be true

$$\begin{aligned} \forall\, x_1, x_2 \in \partial\mathbf{O}_e \quad \text{such that} \quad ||x_1 - x_2|| \leq D_{pp}, \\ x = \lambda x_1 + (1-\lambda)x_2, \quad \lambda \in [0,1], \quad x \notin \mathbf{O}, \end{aligned} \tag{21}$$

where $\mathbf{O}_e$ is a convex set describing the enlarged obstacle, $\partial\mathbf{O}_e$ is its boundary, and $\mathbf{O}$ is a convex set describing the original obstacle.

A simple way to meet the above requirement is by expanding the obstacle by D_{pp}. However, we describe an approach that minimizes the obstacle enlargement, a desired feature from the motion planning point of view. Specifically, given an obstacle described by $OX \leq r$, for every corner c_i of the obstacle described by the two linear inequalities

$$\begin{cases} a^{c_i}x + b^{c_i}y \leq c^{c_i}, & (l_1^{c_i}) \\ d^{c_i}x + e^{c_i}y \leq f^{c_i}, & (l_2^{c_i}) \end{cases}$$

construct the enlarged obstacle by

(i) finding the bisector b^{c_i} of the corner,
(ii) finding the line l^{c_i} orthogonal to b^{c_i} and passing by the vertex of the corner,
(iii) finding the two points $P_1^{c_i}$ and $P_2^{c_i}$ on l^{c_i} with distance $D_{pp}/2$ from the vertex of the corner, and
(iv) finding the lines $e_1^{c_i}$ and $e_2^{c_i}$ respectively passing by $P_1^{c_i}$ and $P_2^{c_i}$ and parallel to $l_1^{c_i}$ and $l_2^{c_i}$. $e_1^{c_i}$ and $e_2^{c_i}$ represent the boundary of the enlargement.

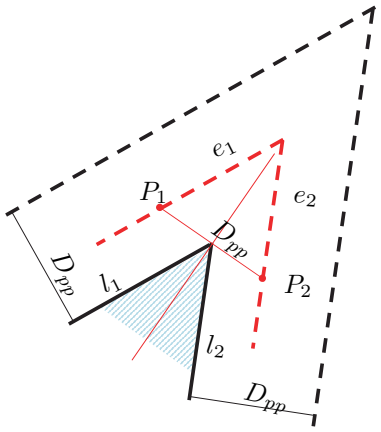


Fig. 2: Enlargement with D_{pp} (in black), and enlargement following the proposed construction (in red).

An example of the construction is shown in Fig. 2. To motivate the choice of this construction, let us consider a $90°$ angle and an enlargement obtained using lines parallel to the original obstacle boundary, see Fig. 3. We need to guarantee that given any two points on the border of the enlargement such that the distance between the two points is D_{pp}, the segment connecting the two points does not intersect the original obstacle. To this end, we show that the segment that passes closer to the corner of an obstacle is the one which is

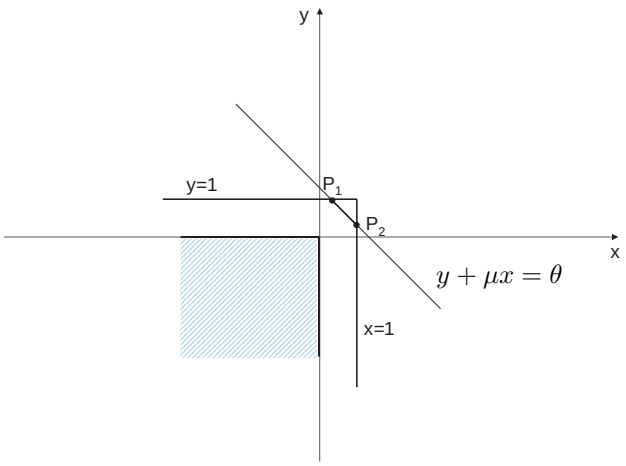


Fig. 3: Example of obstacle expanssion.

orthogonal to the bisector. Given the enlargement in Fig. 3, we characterize the family $y+\mu x=\theta$ of lines that cross the two lines $x=1$ and $y=1$ at the points P_1 and P_2 such that the distance between P_1 and P_2 is $D_{pp}=1$. The general expressions for P_1 and P_2 are

$$P_1=(1,\ \theta-\mu) \qquad P_2=\left(\frac{\theta-1}{\mu},\ 1\right).$$

By adding the requirement that the distance between the two points is D_{pp}, we have

$$\left(1-\frac{\theta-1}{\mu}\right)^2+(\theta-\mu-1)^2=1,$$

and solving for θ yields

$$\theta(\mu)=\frac{\rho+\sqrt{\rho^2-(1+\mu^2)(\rho+\mu+\mu^3+\mu^4)}}{(1+\mu^2)^2},$$

where $\rho \triangleq 1+\mu+\mu^2+\mu^3$. The distance from the corner of the obstacle $(X_0,Y_0)=(0,0)$ and the family of lines $y+\mu x=\theta(\mu)$ is given by

$$dist(\mu)=\frac{Y_0+\mu X_0+\theta(\mu)}{\sqrt{1+\mu^2}}=\frac{\theta(\mu)}{\sqrt{1+\mu^2}}$$

The minimum of the function $dist(\mu)$ is obtained for $\mu=1$, that is the line which is orthogonal to the bisector.

B. Sampling Time

Some considerations about the choice of the sampling period ΔT are in order. In general, it is desirable to have the shortest sampling period so that the discrete-time model is as close as possible to the continuous one. Having a short sampling period would improve the process of avoiding obstacles and collisions. Since between two samples no control action can be applied to recover from an unexpected situation, the shorter the sampling period the better. On the other hand, when an MPC algorithm is used, some other factors need to be taken into account. First of all, to make the optimization problem feasible, the control horizon must be held long enough. Holding the same control horizon and reducing the sampling period would produce more samples in each optimization problem. Hence, more samples imply more variables and in turn a more complex optimization problem to be solved. Therefore, the sampling period can be reduced as far as the resulting optimization problem can still be solved in no more than ΔT.

V. A Centralized Strategy

In this section, a centralized approach to formulate a multirobot target assignment problem is presented. In this case, the MPC algorithm includes all the constraints to formulate a large single optimization problem. This strategy is presented as a basic step toward developing a hierarchical decentralized approach.

A. Open-loop Optimization Problem

The multirobot target assignment problem is summarized as follows

$$\min \sum_{k=0}^{T-1}\sum_{i=1}^{N_R} |u_i^x(k)|+|u_i^y(k)| \tag{22}$$

subject to system dynamics, obstacle avoidance, collision avoidance, and target assignment constraints.

Except for the linear discrete-time dynamics of the system

$$X(k+1)=A\,X(k)+B\,U(k), \qquad \forall k=0,\ldots,T-1, \tag{23}$$

the other constraints and the cost in (22) have to be converted into a mixed integer linear form.

B. Cost Function

The absolute value $|\cdot|$ can be expressed as the composition of linear functions as follows. Let us define an auxiliary variable z_i^x given by

$$\begin{aligned} z_i^x(k) &\geq u_i^x(k), \\ z_i^x(k) &\geq -u_i^x(k). \end{aligned} \tag{24}$$

The two inequalities in (24) model the absolute value of u_i^x. Therefore, the cost function in (22) can be rewritten as

$$\min \sum_{k=0}^{T-1}\sum_{i=1}^{N_R} z_i^x(k)+z_i^y(k).$$

C. Obstacle Avoidance

As stated in the problem formulation, we consider obstacles that can be written as

$$O\begin{pmatrix} x \\ y \end{pmatrix} \leq r, \tag{25}$$

where O is a $R\times 2$ matrix and r is a $R\times 1$ vector. R is the number of constraints necessary to define the obstacle.

Given an obstacle as the one shown in Fig. 4. The matrix describing the obstacle is given by

$$\begin{pmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \leq \begin{pmatrix} x_a \\ -x_b \\ y_a \\ -y_b \end{pmatrix},$$

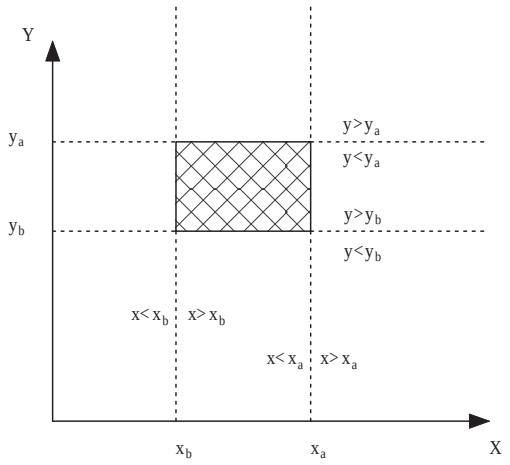


Fig. 4: Obstacle's representation.

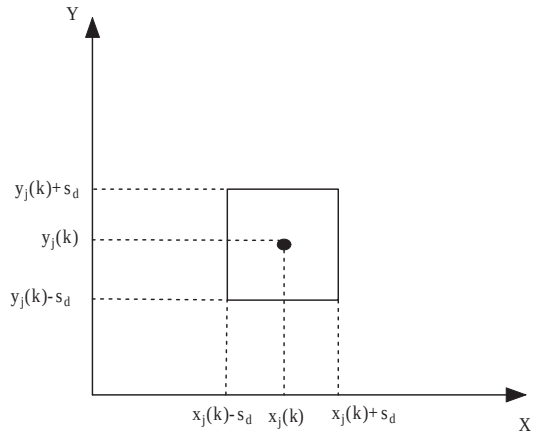


Fig. 5: Robot's safety zone.

A binary auxiliary variable ω_{pi}^k is associated with each constraint that defines the obstacle at every sampling time k. $\omega_{pi}^k = 1$ when the constraint is satisfied by the position of the robot $(x(k), y(k))$. All the points inside the obstacle are characterized by having all the R constraints that define the obstacle satisfied. In contrast, the points outside the obstacle are characterized by having at least one of the constraints violated. Consider robot i at the k-th sample time, it is possible to write

$$o_{p1}x_i(k) + o_{p2}y_i(k) \leq r_p \Rightarrow \omega_{pi}^k = 1, \tag{26a}$$

$$\sum_{p=1}^{R} \omega_{pj}^k \leq R - 1. \tag{26b}$$

Equation (26a) drives the binary auxiliary variable ω_{pi}^k to 1 if the p-th constraint that describes the obstacle is satisfied by the position coordinates of robot i at time k. The robot is inside the obstacle if all the R constraints that describe the obstacle are satisfied. The constraint in (26b) imposes that the coordinates of the robot violate at least one of the constraints that describe the obstacle and ensures that the robot stays outside the obstacle.

The implications in (26a) are translated into mixed integer linear inequalities using the big M technique, thus

$$o_{p1}x_i(k) + o_{p2}y_i(k) - r_p \geq \epsilon + (m - \epsilon)\omega_{pi}^k. \tag{27}$$

Note that the inequalities in (26b) are already in linear form.

D. Collision Avoidance

Collision avoidance could be seen as a special case of obstacle avoidance. A robot sees other robots as moving obstacles to be avoided. To implement collision avoidance a safety zone around the robot is defined and the constraint that no robot can enter the safety zone is added. Fig. 5 depicts the safety zone around the robot.

If we select robot j, its safety zone, defined by a distance s_d, is represented by the following set of inequalities

$$\begin{array}{ll} x \geq x_j(k) - s_d, & x \leq x_j(k) + s_d, \\ y \geq y_j(k) - s_d, & y \leq y_j(k) + s_d, \end{array}$$

that could be written in a more compact matrix form

$$C_A \begin{pmatrix} x \\ y \end{pmatrix} \leq C_A \begin{pmatrix} x_j(k) \\ y_j(k) \end{pmatrix} + \mathbf{s}_d, \tag{28}$$

where

$$C_A = \begin{pmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{pmatrix}, \qquad \mathbf{s}_d = \begin{pmatrix} s_d \\ s_d \\ s_d \\ s_d \end{pmatrix}.$$

As for obstacle avoidance, an auxiliary binary variable $\tau_{ij}^p(k)$ is introduced for each inequality that defines the safety zone. $\tau_{ij}^p(k) = 1$ if the inequality is satisfied. Collision avoidance is obtained by enforcing that the sum of the auxiliary variables $\tau_{ij}^p(k)$ is less than 3 (*i.e.*, number of constraints that define the safety zone minus one). More formally, collision avoidance is modeled with the implication

$$\begin{array}{c} C_A \begin{pmatrix} x_i(k) - x_j(k) \\ y_i(k) - y_j(k) \end{pmatrix} \leq \mathbf{s}_d \Rightarrow \tau_{ij}^p(k) = 1, \\ \sum_{p=1}^{4} \tau_{ij}^p(k) \leq 3, \end{array} \tag{29}$$

where $(x_i(k), y_i(k))$ and $(x_j(k), y_j(k))$ are the position of robot i and j at time k. $\tau_{ij}^p = 1$ if the p-th inequality in the first equation in (29) is satisfied. The implications in (29) are converted into MILP as obstacle avoidance. Note that imposing these constraints is equivalent to requesting that the distance between two robots is at least s_d in the x and y coordinates.

E. Target Assignment

The multirobot target assignment problem addressed in this work consists of assigning one robot to one of the possible targets. The auxiliary variable $\gamma_{ij} = 1$ if robot i is assigned to target j. Clearly, every robot has to be assigned

to one target, and each target needs to be assigned to one robot. From these two observations, we have

$$\sum_{j=1}^{N_R} \gamma_{j\ell} = 1, \quad \forall\, \ell = 1, \ldots, N_T, \tag{30}$$

$$\sum_{\ell=1}^{N_T} \gamma_{j\ell} = 1, \quad \forall\, j = 1, \ldots, N_R, \tag{31}$$

where N_R and N_T are the number of robots and the number of targets, respectively. Therefore, the target assignment implication can be written as

$$\gamma_{j\ell} = 1 \quad \Rightarrow \quad X_j(T) - X_\ell^t = 0, \tag{32}$$

where $X_j(T)$ are the coordinates of robot j at the end of the control horizon T, and X_ℓ^t are the coordinates of target ℓ. The conversion of the implication in (32) into MILP form is not straightforward. First, the implication is expressed into its x and y components, thus

$$\begin{aligned} \gamma_{j\ell} = 1 &\Rightarrow x_j(T) - x_\ell^t = 0, \\ \gamma_{j\ell} = 1 &\Rightarrow y_j(T) - y_\ell^t = 0. \end{aligned} \tag{33}$$

Next, to transform the two implications in (33), more auxiliary variables are needed because the big M technique cannot be applied to implications having equality constraints. The two implications for the x component are

$$\begin{aligned} \gamma_{j\ell} = 1 &\Rightarrow x_j(T) - x_\ell^t \geq 0, \\ \gamma_{j\ell} = 1 &\Rightarrow x_j(T) - x_\ell^t \leq 0. \end{aligned} \tag{34}$$

A similar set of implications are written for y. At this point, all the implications can be translated into MILP form using the big M technique. The resulting set of inequalities are given by

$$x_j(T) - x_\ell^t \geq m(1 - \gamma_{j\ell}), \quad x_j(T) - x_\ell^t \leq M(1 - \gamma_{j\ell}),$$

and

$$y_j(T) - y_\ell^t \geq m(1 - \gamma_{j\ell}), \quad y_j(T) - y_\ell^t \leq M(1 - \gamma_{j\ell}).$$

F. Global Optimization Problem

By collecting all the constraints, the optimization problem to be solved becomes

$$\min \sum_{k=0}^{T-1} \sum_{i=1}^{N_R} z_i^x(k) + z_i^y(k),$$

subject to the dynamic equation of the system

$$\begin{aligned} X_i(k+1) \;&=\; A_i\, X_i(k) + B_i\, U_i(k), \\ &\quad \forall\, k = 0, \ldots, T-1,\ i = 1, \ldots, N_R, \end{aligned}$$

the equations that define $z_i^{x|y}$

$$\begin{aligned} &z_i^{x|y}(k) \geq u_i^{x|y}(k) \\ &z_i^{x|y}(k) \geq -u_i^{x|y}(k),\ k = 0, \ldots, T-1;\ i = 1, \ldots, N_R, \end{aligned}$$

the obstacle avoidance constraints

$$\begin{aligned} &o_{p1} x_i(k) + o_{p2} y_i(k) - r_p \geq \epsilon + (m - \epsilon)\omega_{pi}^k, \\ &\sum_{p=1}^{R} \omega_{pi}^k \leq R - 1 \qquad \forall k = 0, \ldots, T-1; i = 1, \ldots, N_R, \end{aligned}$$

the collision avoidance constraints

$$\begin{aligned} &ca_{p1}(x_i(k) - x_j(k)) + ca_{p2}(y_i(k) - y_j(k)), \\ &\qquad\qquad -s_d \geq \epsilon + (m - \epsilon)\tau_{pij}^k, \\ &\sum_{p=1}^{4} \tau_{pij}^k \leq 3, \\ &\forall\, k = 0, \ldots, T-1,\ i = 1, \ldots, N_R,\ j = i+1, \ldots, N_R, \end{aligned}$$

the target assignment constraints

$$\begin{aligned} &\sum_{j=1}^{N_R} \gamma_{j\ell} = 1, && \forall\, \ell = 1, \ldots, N_T, \\ &\sum_{\ell=1}^{N_T} \gamma_{j\ell} = 1, && \forall\, j = 1, \ldots, N_R, \\ &x_i(T) - x_\ell^t \geq m(1 - \gamma_{j\ell}), && x_i(T) - x_\ell^t \leq M(1 - \gamma_{j\ell}), \\ &y_i(T) - y_\ell^t \geq m(1 - \gamma_{j\ell}), && y_i(T) - y_\ell^t \leq M(1 - \gamma_{j\ell}), \\ &\forall\, i = 1, \ldots, N_R;\ \ell = 1, \ldots, N_T, \end{aligned}$$

and, the bounds on the input and robot's velocity

$$u_i^{x|y}(k) \leq U_{\max}, \quad v_i^{x|y}(k) \leq V_{\max}, \quad \forall\, i = 1, \ldots, N_R.$$

This optimization problem requires $4N_RT$ continuous variables to describe the state of each robot during the entire control horizon T, $2N_RT$ variables for the control input applied to each robot, $2N_RT$ auxiliary variables $z_i^{x|y}$, $N_RT\sum_{i+1}^{N_O} R_i$ binary variables for obstacle avoidance, $4T\sum_{i=1}^{N_R} i$ binary variables for collision avoidance, and N_RN_T variables for target assignment. Additionally, from the point of view of the number of constraints present in the problem, the dynamic equations require $4N_RT$, the definition of $z_i^{x|y}$ needs $2N_RT$ constraints, obstacle avoidance requires $N_RT\sum_{i+1}^{N_O}(R_i + 1)$, collision avoidance brings $5T\sum_{i=1}^{N_R} i$ constraints, target assignment gives $4N_RT + N_R + N_T$ constraints, and finally the bounds on the input and velocity require $4N_RT$ constraints.

This analysis of the global centralized algorithm should give an idea of the dimensionality of the optimization problem. Even for a small team and simple environment (few obstacles), the number of variables and constraints can total in the thousands, making the optimization problem computationally intractable. The need of scalable and more efficient approaches to solve optimization problems involving cooperative robots, motivates the strategies described in the next section.

VI. A Hierarchical and Decentralized Strategy

When a problem is large and complex to be solved in a single optimization stage, a common practice is to replace the original problem with smaller subproblems. Examples of

hierarchical decision making are everywhere. For instance, in a manufacturing plant the higher-level planner takes decisions based on the market's demand about which part to produce and its quantity. Then, a command is sent to a lower-level controller that organizes the machines to meet the required production output. An agent at the lower-level does not need to know the overall strategy or plan for all the other agents to accomplish a task. It probably needs to know only what its neighbors are doing (*e.g.*, if a machine in a supply chain is going to increase the production rate, it will need to verify that the previous one can follow the new production rate).

In our hierarchical formulation shown in Fig. 6, a higher-level planner makes decisions related to the whole team such as target assignment and formation keeping. On the other hand, the lower-level controller which is local to each agent handles obstacle and collision avoidance. Instead of planning an off-line trajectory, the higher-level planner periodically reformulates and solves an optimization problem (*i.e.*, MPC) to compensate for uncertainty or new information available in the environment. Furthermore, the lower-level controller is decentralized in the sense that every robot solves its own optimization problem to find a safe trajectory. The local optimization problem considers only the robot's closest obstacles and teammates. We assume that each robot has a limited sensing range. Therefore, only obstacles and neighbors within a robot's sensing range are considered in the optimization, see Fig. 7. Consequently, the solution is necessarily suboptimal.

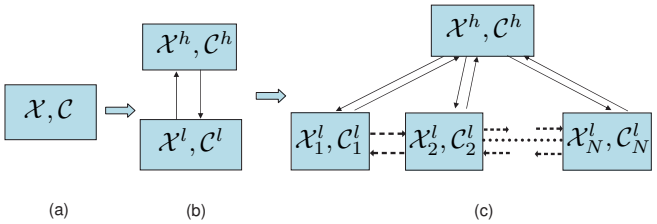


Fig. 6: From a large problem (a) to the hierarchical formulation (b) to the hierarchical/decentralized one (c). Note that full arrows represent communication links, while dotted arrows represent sensing links.

The next sections discuss some heuristics for improving the performance of the hierarchical/decentralized strategy. Roughly speaking, the higher-level planner solves an open-loop optimization problem to find the best global robot/target assignment. The task of the lower-level controller is to drive a robot to the assigned target while avoiding collisions with obstacles and neighbors.

A. Higher-level Optimization Problem

The higher-level MPC algorithm is in charge of finding the best robot/target assignment based on the current state of the team. The position of the assigned target j is communicated to robot i. Specifically, the higher-level MPC solves the

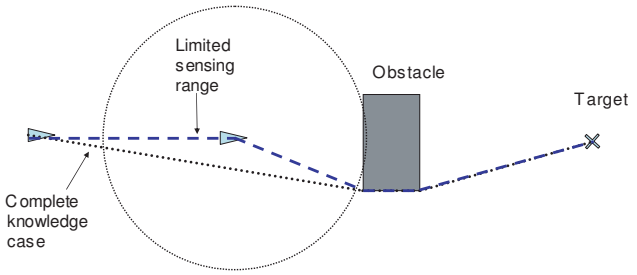


Fig. 7: The difference in the trajectories in the case of complete knowledge and partial knowledge is one of the reasons the solution found with the decentralized method is worse than the one found with the global optimization.

following open-loop optimization problem

$$\min \sum_{i=1}^{N_R} \sum_{j=1}^{N_T} c_{ij} \tau_{ij}, \tag{36a}$$

subject to

$$\sum_{i=1}^{N_R} \gamma_{ij} = 1, \qquad \forall\, j = 1, \ldots, N_T, \tag{36b}$$

$$\sum_{j=1}^{N_T} \gamma_{ij} = 1, \qquad \forall\, i = 1, \ldots, N_R, \tag{36c}$$

where c_{ij} is the distance between robot i and target j. The variable γ_{ij} is a binary variable that is one if robot i is assigned to target j or zero otherwise. This assignment problem can be formulated as a linear program that can be efficiently solved by standard methods.

B. Lower-level Optimization Problem

The lower-level optimization problem computes the control inputs that drive a robot to its assigned target while avoiding collisions. The optimization problem for robot i is given by

$$\mathcal{P}_i^L = \min \sum_{k=0}^{T-1} z_i^x(k) + z_i^y(k), \tag{37}$$

subject to the equations that define $z_i^{x|y}(k)$

$$\begin{aligned} &z_i^{x|y}(k) \geq u_i^{x|y}(k), \\ &z_i^{x|y}(k) \geq -u_i^{x|y}(k), \qquad \forall\, k = 0, \ldots, T-1, \end{aligned} \tag{38}$$

the dynamic equations of a robot

$$\begin{aligned} &X_i(k+1) = A_i X_i(k) + B_i U_i(k), \\ &X_i(k) = \begin{bmatrix} x_i(k) \\ y_i(k) \\ v_i^x(k) \\ v_i^y(k) \end{bmatrix}, \; U_i(k) = \begin{bmatrix} u_i^x(k) \\ u_i^y(k) \end{bmatrix}, \quad \forall\, k = 0, \ldots, T-1, \end{aligned} \tag{39}$$

the constraints for obstacle avoidance

$$\begin{aligned} &o_{p1}x_i(k) + o_{p2}y_i(k) - r_p \geq \epsilon + (m-\epsilon)\omega_{pi}^k, \\ &\sum_{p=1}^{R} \omega_{pi}^k \leq R-1 \qquad \forall k = 0,\ldots,T-1, \end{aligned} \tag{40}$$

the constraints for inter-robot collision avoidance

$$\begin{aligned} &ca_{p1}(x_i(k) - \hat{x}_j(k)) + ca_{p2}(y_i(k) - \hat{y}_j(k)) \\ &\qquad - s_d \geq \epsilon + (m-\epsilon)\tau_{pij}^k, \\ &\sum_{p=1}^{4} \tau_{pij}^k \leq 3 \qquad \forall\, k = 0,\ldots,T-1;\ j = 1,\ldots,N_N, \end{aligned} \tag{41}$$

where N_N is the number of neighbors, the terminal set constraints

$$x_i(T) = x_j^T, \quad y_i(T) = y_j^T, \tag{42}$$

and, the bounds on inputs and robot's velocities

$$u_i^{x|y}(k) \leq U_{\max},\ v_i^{x|y}(k) \leq V_{\max}, \quad \forall\, k = 1,\ldots,T. \tag{43}$$

Note that the constraints in (40) cannot be written in the same way as for the global optimization case. In the global formulation, the variables $x_j(k)$ and $y_j(k)$ are part of the optimization problem. However, these variables are unknown for the hierarchical/decentralized algorithm. Therefore, a robot should be able to estimate the position of its neighbors $(\hat{x}_j(k),\ \hat{y}_j(k))$ by communication or sensing. We use the latter, since synchronous communications would require a complex network structure. We assume that each robot has the ability to sense the position and velocity of its neighbors (*e.g.*, using an omnidirectional camera [12]). The estimated positions are given by

$$\begin{aligned} \hat{x}_j(k) &= x_j(0) + kv_{xj}(0), \\ \hat{y}_j(k) &= y_j(0) + kv_{yj}(0). \end{aligned}$$

Since most likely robot j would change its velocity to ensure a collision free solution, all the reachable points of robot j will be made unaccessible to robot i. This strategy is equivalent to enlarging the safety zone around robot j by $s_d = V_{\max}\Delta T^L$ where ΔT^L is the lower-level sampling time. Although conservative, this approach is necessary to ensure collision avoidance.

The variables and constraints needed in the problem are: $2T$ continuous variables and $4T$ constraints for the definition of $z_i^{x|y}(k)$ (38), $6T$ continuous variables and $4T$ constraints for the dynamic equations (39), $\sum_i^{N_o} R_i T$ binary variables and $\sum_i^{N_o}(R_i+1)T$ constraints due to the obstacle avoidance (40) where N_o is the number of obstacles, and $4TN_N$ binary variables and $5TN_N$ constraints for the collision avoidance (41). As it can be seen, the dimension of the hierarchical/decentralized problem is much smaller than the global one. Moreover, the number of binary variables depends on the number of obstacles considered N_o, and the number of neighbors N_N. The next section describes some heuristic to further reduce the dimensionality of the problem.

C. Heuristics for the Lower-level Optimization

The optimization problem $\mathcal{P}_i^L$ is defined through the set of variables $\mathcal{X}_i^L$ and the set of constraints $\mathcal{C}_i^L$. The main objective is to reduce the cardinality of these two sets as much as possible which in turn reduces the computation time required to solve the problem. Also, it is desirable to keep the solution collision free and as close as possible to the global optimum.

Among all the variables and constraints in the problem, there is a subset of variables $\mathcal{X}_i^n \subset \mathcal{X}_i^L$ and constraints $\mathcal{C}_i^n \subset \mathcal{C}_i^L$ which are required. These variables and constraints correspond to the definition of $z_i^{x|y}(k)$ (38), the dynamic equations of robots (39), and the bounds on state variables and inputs (43).

Let us denote the set of variables and constraints needed to include an obstacle $\mathbf{O}_t$ respectively by $\mathcal{X}_t^o$ and $\mathcal{C}_t^o$. Similarly, the variables and constraints for including robot j are $\mathcal{X}_j^r$ and $\mathcal{C}_j^r$. One way to build the two sets $\mathcal{X}_i^L$ and $\mathcal{C}_i^L$ is by including all the obstacles and robots in the network. This means the two sets are given by

$$\begin{aligned} \mathcal{X}_i^L &= \mathcal{X}_i^n \bigcup_{t=1}^{N_o} \mathcal{X}_t^o \bigcup_{j=1}^{N_R} \mathcal{X}_j^r, \\ \mathcal{C}_i^L &= \mathcal{C}_i^n \bigcup_{t=1}^{N_o} \mathcal{C}_t^o \bigcup_{j=1}^{N_R} \mathcal{C}_j^r. \end{aligned}$$

To define the above sets complete knowledge of the environment and state of the robotic network is required. Although the solution of this optimization problem would be close to the global optimum, it has several drawbacks. Since the two sets $\mathcal{X}_i^L$ and $\mathcal{C}_i^L$ are the largest possible variable and constraint sets, computing a solution to the problem would require a long period of time. Moreover, complete knowledge of the environment and position and velocity of all the members in the team might not be possible.

An alternative approach is to include in the sets $\mathcal{X}_i^L$ and $\mathcal{C}_i^L$ only the robots and obstacles that can be sensed as shown in Fig 8(a). Following this idea the variables and constraints

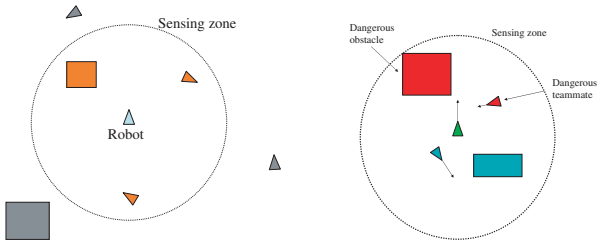


Fig. 8: (a) A robot can *see* only the obstacles and neighbors inside its sensing zone. Everything outside this zone is unknown. (b) For the green robot only the red obstacle and robot represent threats.

of an obstacle $\mathbf{O}_t$ are included if

$$\mathcal{X}_t^o \in \mathcal{X}_i^L,\ \mathcal{C}_t^o \in \mathcal{C}_i^L \Leftrightarrow \min_{(x,y)\in\mathbf{O}_t} d([x_i(k), y_i(k)], [x, y]) \leq d_s.$$

Also, the variables and constraints of collision avoidance with robot j are included if

$$\mathcal{X}_j^r \in \mathcal{X}_i^L,\ \mathcal{C}_j^r \in \mathcal{C}_i^L \Leftrightarrow d([x_i(k), y_i(k)], [x_j(k), y_j(k)]) \leq d_s.$$

An assumption on the sensing range d_s is in order. To guarantee the existence of a collision free trajectory, the sensing range must be large enough to allow a robot to stop before hitting the obstacles when moving at maximum speed. This means that in the time t_s required for the robot to stop while moving at its maximum velocity

$$V_{\max} - t_s U_{\max} = 0,$$
$$t_s = \frac{V_{\max}}{U_{\max}},$$

the distance covered x should be less than d_s, thus

$$x = V_{\max} t_s - U_{\max} \frac{t_s^2}{2} \leq d_s,$$
$$x = \frac{1}{2} \frac{V_{\max}^2}{U_{\max}} \leq d_s.$$

Using this heuristic, the number of obstacles and neighbors considered in each lower-level optimization problem are clearly reduced. Since the neglected obstacles and teammates are far from the robot's actual position, there is no risk of collision. The result obtained with these heuristics can be further improved. Although obstacles or teammates are in the sensing range of robot i, they may not represent a risk. Let us consider, for example, robot j in the robot i sensing zone. If j is behind i and moving in the opposite direction, j is not a threat for i. Moreover, if an obstacle is in the sensing range but it is behind robot i, with i moving away from the obstacle, then there is no need to include the obstacle in the optimization problem. See Fig. 8(b).

Considering the actual velocity and the bound on the acceleration, it is possible to build the reachability cone as depicted in Fig. 9. This cone can be used to estimate the position of a robot in the next sampling time. The idea is to use this cone to decide which robot and which obstacle are effectively dangerous and need to be included in the optimization problem.

Given the velocity of robot i at the k-th sampling time $(V_x(k),\ V_y(k))$, the maximum variation possible in a sampling time is $\pm U_{\max}\Delta T$ for each component. The following four vectors can be obtained

$$v_1 = \begin{pmatrix} V_x + U_{\max}\Delta T \\ V_y + U_{\max}\Delta T \end{pmatrix}, \quad v_2 = \begin{pmatrix} V_x + U_{\max}\Delta T \\ V_y - U_{\max}\Delta T \end{pmatrix},$$
$$v_3 = \begin{pmatrix} V_x - U_{\max}\Delta T \\ V_y + U_{\max}\Delta T \end{pmatrix}, \quad v_4 = \begin{pmatrix} V_x - U_{\max}\Delta T \\ V_y - U_{\max}\Delta T \end{pmatrix}.$$

Among these, two of them v_i and v_j can be obtained as cone combination of the other two v_k and v_p. The latter two are taken to generate the cone that will be used as estimation of the future position of the robot. Specifically, the cone is defined by

$$\mathbf{C}_i = \{(x, y) \in \mathbb{R}^2 : (x, y) = \lambda_k v_k + \lambda_p v_p, \quad \lambda_k, \lambda_p \geq 0\}.$$

Once the cone $\mathbf{C}$ is obtained, the intersection between the cone and an obstacle $\mathbf{O}_t$ can be written as

$$\mathbf{I}_o = \mathbf{O}_t \bigcap \mathbf{C}_i.$$

If the intersection is empty, the obstacle is not on the way and can be omitted. If the intersection is not empty, the obstacle is included, provided that the distance between the closest intersection point and the position of robot i is less than a threshold σ, that is

$$\mathcal{C}_t^o \in \mathcal{C}_i^L, \text{ and } \mathcal{X}_t^o \in \mathcal{X}_i^L,$$
$$\Updownarrow$$
$$\mathbf{I}_o \neq \emptyset \bigwedge \min_{(x,y)\in\mathbf{I}_o} d([x_i(k), y_i(k)], [x, y]) \leq \sigma.$$

Now, to decide if robot j must be included in the optimization problem of robot i, the intersection $\mathbf{I}_r$ between the cone $\mathbf{C}_i$ and $\mathbf{C}_j$ is computed

$$\mathbf{I}_r = \mathbf{C}_i \bigcap \mathbf{C}_j.$$

In case the intersection is empty, robot j is not on the way and can be omitted. However, if the intersection is not empty, robot j is included, provided that the distance between the closest intersection point and the position of robot i is less than σ. Summarizing, we have

$$\mathcal{C}_j^r \in \mathcal{C}_i^L, \text{ and } \mathcal{X}_j^r \in \mathcal{X}_i^L,$$
$$\Updownarrow$$
$$\mathbf{I}_r \neq \emptyset \bigwedge \min_{(x,y)\in\mathbf{I}_r} d([x_i(k), y_i(k)], [x, y]) \leq \sigma.$$

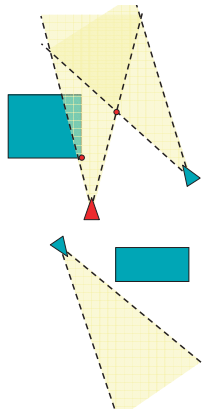
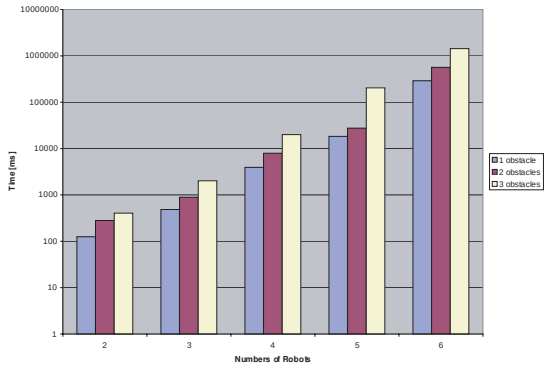


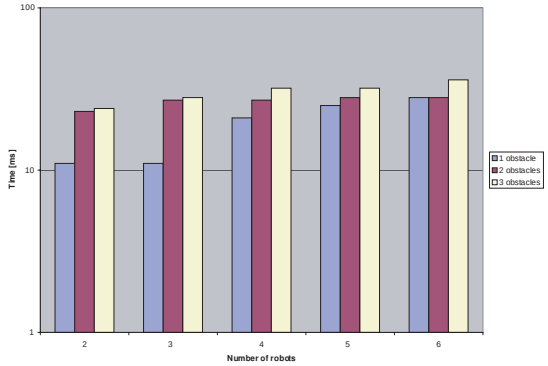
Fig. 9: Among the obstacles and neighbors sensed by the red robot only a subset has non empty intersection with its cone. Note that the red dots are the closest points of the intersections.

D. More Targets than Agents

Let us assume we have a limited number of agents to cover a large number of targets. This problem can be easily solved by adding few modifications to the previous formulation. In this case, it is not possible to have a one to one mapping agent-target. Therefore, every agent may have to visit more than one target. A scheduling problem must be solved to find which targets each agent must visit, and in which order. It is well known that scheduling is an NP-hard problem. Thus,



(a) Global case.



(b) Hierarchical/distributed case.

Fig. 10: Average time required to solve one optimization problem as a function of the number of obstacles and agents in a semi-logarithmic scale.

to avoid computational problems, a sub-optimal algorithm can be used instead. The sub-optimal algorithm will solve an assignment problem over the set $\mathcal{T}$. Once a target t_i is reached, it is removed from the original set of targets $\mathcal{T}$. In this way the set $\mathcal{T}$ is reduced every time a target is reached, until its cardinality decreases to the number of agents. To address the problem as described, the equations used for the target assignment must be modified. In particular, the constraint that at each target must be assigned an agent, should changed. In fact, since at the beginning the number of targets is greater than the number of agents, the initial assignment will have some of the targets not covered by any agent. For that reason, equation (36b) needs to be substituted with

$$\sum_{i=1}^{N_R} \gamma_{ij} \leq 1, \qquad \forall\, j = 1, \ldots, N_T, \tag{44}$$

such that the '$\leq$' instead of '$=$' will permit some targets having no agent assigned to them.

The reader is referred to our previous work in [17] for more examples of modeling of different scenarios using the MPC/MILP framework.

VII. Analysis of Results

The global and hierarchical/decentralized algorithms are coded in Matlab. To solve the MILP problem the commercial solver CPLEX [1] is used. CPLEX functions are called from Matlab through an interface described in [3]. To compare the global version and the hierarchical/decentralized approach some simulations are presented. In particular, we consider scenarios with one to three obstacles and with two to six agents. The average time required to solve a single optimization problem is reported in Fig. 10. Comparing the scenario with six robots and three obstacles, the centralized algorithm requires 1433 s while the hierarchical/decentralized algorithm takes only 0.036 s. As it can be seen the computation time is drastically reduced in the hierarchical/decentralized case. However, such an improvement in the computation time comes with a deterioration of the solution. Fig. 11 gives a comparison between the costs of the global and hierarchical/decentralized solutions. As expected, the centralized algorithm is consistently better than the hierarchical/decentralized one. The cost of the hierarchical/decentralized solution is about 2.2-2.5 times the cost of the centralized solution.

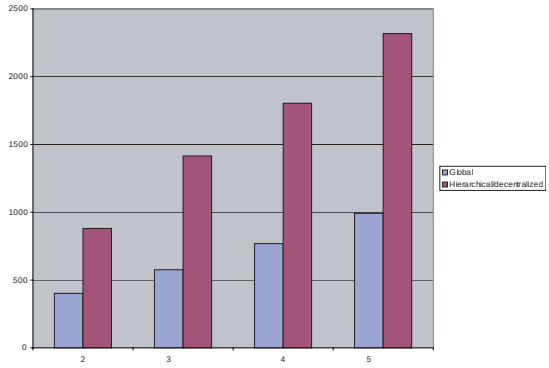


Fig. 11: Comparison between the cost of the global solution and the cost of the hierarchical/decentralized solution.

Fig. 12 depicts a simulation of six robots moving toward six targets in an environment with several obstacles of different shapes. The original planned trajectory (*i.e.*, straight line) of the robots changes when obstacles are detected. A more realistic situation is shown in Fig. 13. In this example, three agents are moving in an urban environment (OSU campus) with the purpose of visiting six view points. Velocities and accelerations, and inter-robot distances are shown in Fig. 14 and Fig. 15, respectively.

Finally, to verify the scalability of the hierarchical/decentralized approach, a target assignment involving 100 robots is illustrated in Fig. 16.

VIII. Conclusions

In this paper we present a hierarchical/decentralized strategy for coordinating robot networks. The algorithm is based on an MPC control loop using MILP optimization techniques. Although, MILP provides flexibility to model cooperative tasks and environments, its drawback lies in its NP-hard nature. The developed hierarchical, decentralized structure allows to replace a large computationally intractable problem into smaller problems that can be solved in a reasonable amount of time. To improve the efficiency of the algorithm

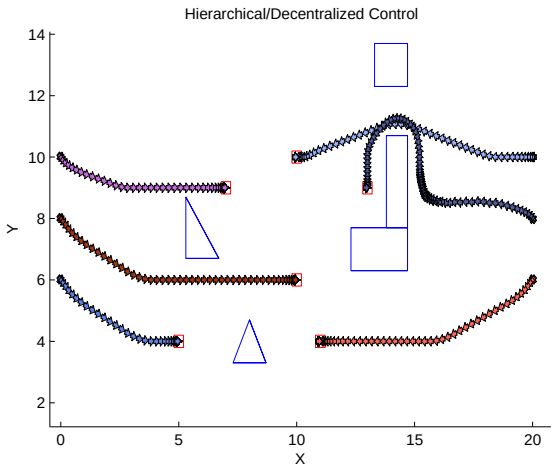


Fig. 12: Six robots going to six targets while avoiding obstacles.

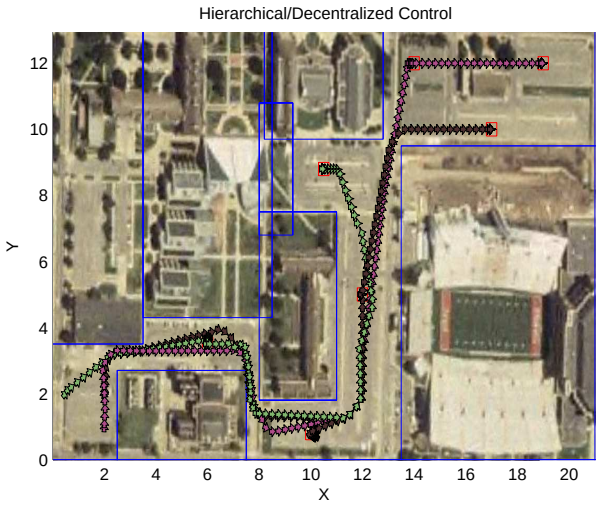


Fig. 13: Three agents visiting view points in a urban environment.

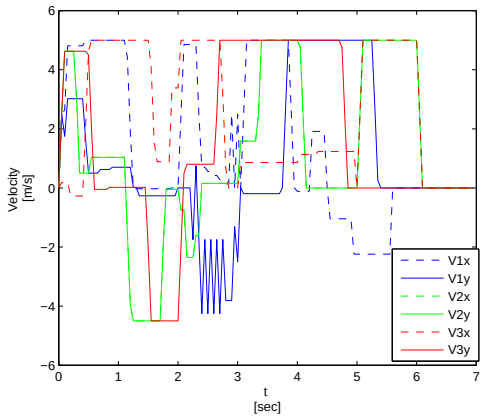
some heuristics are also presented. These heuristics allow us to drastically reduce the number of constraints and binary variables in the optimization problem. Finally, several scenarios are simulated to showcase the flexibility and scalability of the method.

ACKNOWLEDGEMENTS

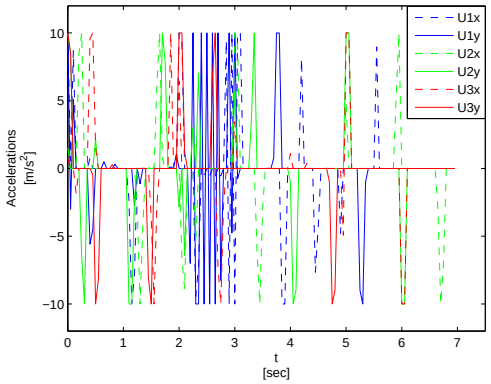
The authors would like to thank Prof. John Spletzer of Lehigh University for useful discussions on optimal multi-robot target assignment.

REFERENCES

[1] *ILOG Cplex: User's manual*. ILOG, 2003.
[2] M. Alighanbari and J. P. How, "Cooperative task assignment of unmanned aerial vehicles in adversarial environments," in *Proceedings of the American Control Conference*, Portland, Oregon, June 8-10 2005, pp. 4661–4666.
[3] M. Baotic, *Matlab interface to CPLEX*. Available from http://control.ee.ethz.ch/ hybrid/cplexint.php.



(a)



(b)

Fig. 14: Velocities and accelerations of the agents.

[4] C. Belta and V. Kumar, "Optimal motion generation for groups of robots: a geometric approach," *ASME Journal of Mechanical Design*, vol. 126, pp. 63–70, 2004.
[5] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, March 1999.
[6] M. R. Benjamin, "Multi-objective navigation and control using interval programming," in *Multi-Robot Systems: From Swarms to Intelligent Automata, Volume II*, A. C. Shultz, L. E. Parker, and F. E. Schneider, Eds. Kluwer Academic Publishers, March 2003.
[7] R. R. Bitmead, M. Gevers, and V. Wertz, *Adaptive optimal control - The thinking man's GPC*. Englewood Cliffs, NJ: Prentice-Hall, 1990.
[8] S. Boyd and L. Vandenberghe, *Communications, Computation, Control and Signal Processing: a Tribute to Thomas Kailath*. Kluwer, 1997, ch. Semidefinite programming relaxations of non-convex problems in control and combinatorial analysis.
[9] ——, *Convex Optimization*. Cambridge, UK: Cambridge University Press, March 2004.
[10] G. C. Chasparis and J. S. Shamma, "Linear-programming-based multi-vehicle path planning with adversaries," in *Proceedings of the American Control Conference*, Portland, Oregon, June 8-10 2005, pp. 1072–1077.
[11] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, April 2004.
[12] A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, and C. J. Taylor, "A vision-based formation control framework," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 813–825, October 2002.
[13] I. Dryden and K. Mardia, *Statistical Shape Analysis*. John Wiley and Sons, 1998.

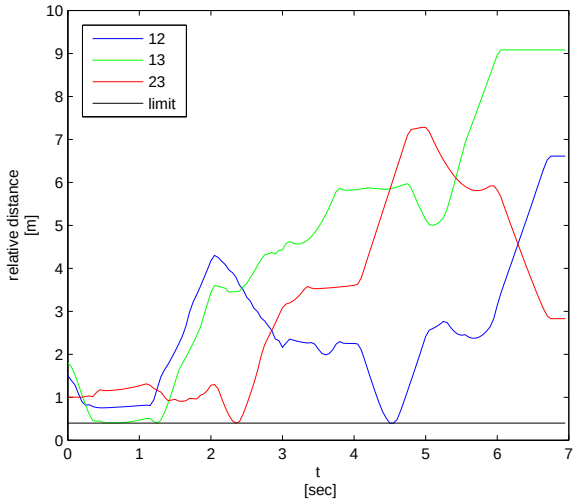


Fig. 15: Relative distances among agents.

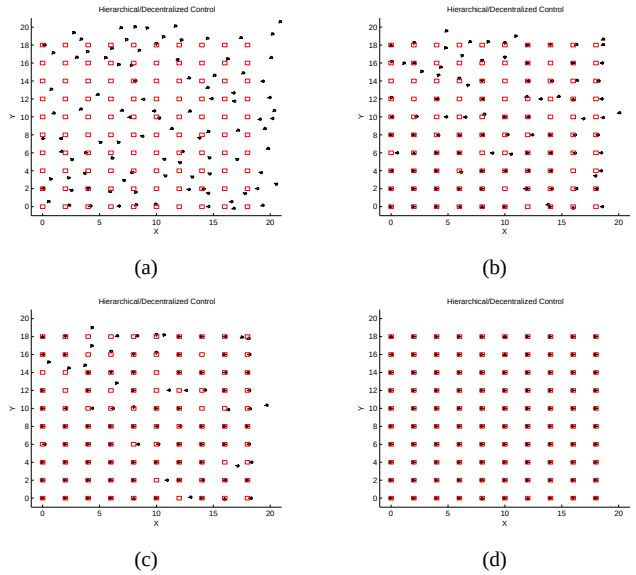


Fig. 16: Snapshots of a simulation with 100 agents moving toward 100 targets.

[14] W. B. Dunbar and R. M. Murray, "Distributed receding horizon control with application to multi-vehicle formation stabilization," *Automatica*, vol. 2, no. 4, pp. 549–558, 2006.

[15] M. G. Earl and R. D'andrea, "A decomposition approach to multi-vehicle cooperative control," Department of Mechanical and Aerospace Engineering Cornell University, Tech. Rep., 2004, available at http://control.mae.cornell.edu/earl/.

[16] ——, "Iterative milp methods for vehicle control problems," in *Proceedings of the IEEE Conference on Decision and Control*, vol. 4, Atlantis, Paradise Island, Bahamas, December 14-17 2004, pp. 4369–4374.

[17] R. Fierro, C. Branca, and J. Spletzer, "On-line optimization-based coordination of multiple unmanned vehicles," in *IEEE Int. Conference on Networking, Sensing, and Control*, Tucson, AZ, March 19-22 2005, pp. 716–721.

[18] R. Fierro and K. Wesselowski, "Optimization-based control of multi-vehicle systems," in *Cooperative Control*, ser. LNCIS, V. Kumar, N. E. Leonard, and A. S. Morse, Eds. Berlin: Springer, 2005, vol. 309, pp. 63–78.

[19] V. Gazi and K. M. Passino, "Stability analysis of swarms," *IEEE Trans. on Automatic Control*, vol. 48, no. 4, pp. 692–697, 2003.

[20] Y. Hao, A. Davari, and A. Manesh, "Trajectory planning for multiple unmanned aerial vehicles using differential flatness and mixed-integer linear programming," *Submitted to Journal of Robotics and Autonomous Systems*, 2005.

[21] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Trans. on Automatic Control*, vol. 48, no. 6, pp. 988–1001, June 2003.

[22] S. S. Keerthi and E. Gilbert, "Optimal, infinite horizon feedback law for a general class of constrained discrete system: Stability and moving-horizon approximations." *Journal of Optimization Theory and Application*, vol. 57, pp. 265–293, 1988.

[23] T. Keviczky, F. Borrelli, and G. J. Balas, "A study on decentralized reciding horizon control for decoupled system," vol. 6, Boston, Massachusset, June 30 - July 2 2004, pp. 4921–4926.

[24] D. B. Kingston and C. J. Schumacher, "Time-dependent cooperative assignment," in *Proceedings of the American Control Conference*, Portland, Oregon, June 8-10 2005, pp. 4084–4089.

[25] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE Int. Conf. on Robotics and Automation*, San Francisco, CA, April 2000, pp. 95–101.

[26] S. M. LaValle and S. A. Hutchinson, "Optimal motion planning for multiple robots having independent goals," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 912–925, Dec. 1998.

[27] Y. Liu, J. B. Cruz, and A. G. Sparks, "Coordinated networked uninhabited air vehicles for persistent area denial," in *Proceedings of the IEEE Conference on Decision and Control*, Paradise Island, Bahamas, December 2004, pp. 3351–3356.

[28] S. Martinez and F. Bullo, "Optimal sensor placement and motion coordination for target tracking," *Automatica*, vol. 42, no. 4, pp. 661–668, 2006.

[29] D. Q. Mayne, J. B. Rawings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, June 2000.

[30] N. Moshtagh and A. Jadbabaie, "Distributed geodesic control laws for flocking of nonholonomic agents," *IEEE Trans. on Automatic Control*, 2006, to appear.

[31] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Trans. on Automatic Control*, vol. 51, no. 3, pp. 401–420, March 2006.

[32] R. L. Raffard, C. J. Tomlin, and S. Boyd, "Distributed optimization for cooperative agents: Application to formation flight," in *Proceedings of the IEEE Conference on Decision and Control*, vol. 3, Atlantis, Paradise Island, Bahamas, December 14-17 2004, pp. 2453 – 2459.

[33] A. Richards and J. P. How, "A decentralized algorithm for robust constrained model predictive control," in *Proceedings of the American Control Conference*, Boston, Massachusetts, June 30-July 2 2004, pp. 4261–4266.

[34] ——, "Mixed-integer programming for control," in *Proceedings of the American Control Conference*, Portland, Oregon, June 8-10 2005, pp. 2676–2683.

[35] D. H. Shim, H. J. Kim, and S. Sastry, "Decentralized nonlinear model predictive control of multiple flying robots," in *Proceedings of the IEEE Conference on Decision and Control*, vol. 4, Maui, Hawaii, December 9-12 2003, pp. 3621–3626.

[36] T. Shima, S. J. Rasmussen, and A. Sparks, "Uav cooperative multiple task assignment using genetic algorithms," in *Proceedings of the American Control Conference*, Portland, Oregon, June 8-10 2005, pp. 2989–2994.

[37] T. Shima, S. J. Rasmussen, A. Sparks, and K. Passino, "Multiple task assignments for cooperating uninhabited aerial vehicles using genetic algorithms," *Computers & Operations Research*, vol. 33, no. 11, pp. 3252–3269, 2005.

[38] J. Spletzer and R. Fierro, "Optimal position strategies for shape changes in robot teams," in *IEEE Int. Conf. on Robotics and Automation*, Barcelona, Spain, April 18-22 2005, pp. 754–759.

[39] T. Vicsek, A. Czirok, E. B. Jacob, I. Cohen, and O. Schochet, "Novel type of phase transitions in a system of self-driven particles," in *Physical Review Letters*, vol. 75, 1995, pp. 1226–1229.

[40] H. P. Williams, *Model Building in Mathematical Programming*. New York: John Wiley & Sons, 1985.