

On-line Optimization-based Coordination of Multiple Unmanned Vehicles

Rafael Fierro and Carlo Branca
School of Electrical & Computer Engineering
Oklahoma State University
Stillwater, OK 74078, USA
{rfierro, carlo.branca}@okstate.edu

John R. Spletzer
Department of Computer Science & Engineering
Lehigh University
Bethlehem, PA 18015, USA
spletzer@cse.lehigh.edu

Abstract—The objective of this work is to investigate on-line optimization-based coordination strategies for robot teams to efficiently accomplish a mission (*e.g.*, reach a set of assigned targets) while avoiding collisions. The multi-robot coordination problem is addressed by solving an on-line receding-horizon mixed-integer program to find some suitable inputs for the vehicles. Simulations results verify the feasibility of our approach.

I. INTRODUCTION

Over the last several years, there has been a great deal of interest in developing cooperative mobile robots for applications such as search and rescue, homeland security, and environmental monitoring to mention just a few. One of the major driving factors behind this interest is the new robotic applications originating from both the military and the civilian domains. In these roles, tasks may be extremely difficult for a single robot to accomplish. Thus, a system composed of teams of cooperative robots is desirable because of its flexibility and fault tolerance.

Several authors have addressed the coordination problem of multiple unmanned vehicles using optimization techniques. Contributions in this area include the work in [1], where the focus is on autonomous vehicles performing distributed sensing tasks. Decentralized optimization based control algorithms are developed in [2] to solve a variety of multi-robot problems. Optimal motion planning is considered in [3], [4]. More recently, the use of model predictive control (MPC) or receding-horizon control (RHC) [5], [6] is becoming popular in the multi-robot system literature [7], [8], [9], [10], [11].

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.

Mixed integer linear programming (MILP) allows the encoding of logical rules, decisions and constraints into the optimization problem [12]. This capability makes MILP an attractive tool for solving a class of multi-vehicle coordination problems. Maintaining a minimum inter-vehicle distance and avoiding obstacles in the environment are common requirements in multi-robot systems [13], [14]. These are *non-convex* constraints which make the coordination problem very difficult to solve using optimization techniques. Fortunately, current MILP solvers (*e.g.*, CPLEX [15]) can handle non-convex constraints reasonably well.

The main idea in this paper is to combine MPC and MILP into an on-line optimization framework capable of solving a class of coordination problems in multi-vehicle systems. Specifically, we leverage the ability of MPC to incorporate motion constraints and changes in mission objectives, as well as the capability of MILP to solve optimization problems including logical possibly non-convex constraints that naturally model the behavior of multiple robots interacting with each other within a dynamic environment.

In [14], the integration of MPC and MILP has been experimentally demonstrated in trajectory generation and planning for UAV coordination. In this paper however, we focus on investigating the feasibility of MPC/MILP to solve coordination problems ranging from target assignment, to collision and threat avoidance, to formation control. Most importantly, the limitations of MPC/MILP and possible approaches to overcome or at least alleviate these limitations are outlined herein.

II. PRELIMINARIES

A. Model Predictive Control

Model predictive control (MPC) consists of solving an on-line finite horizon open-loop optimization problem using the current state as the initial state for the plant. The solution to the problem gives a sequence of control inputs for the entire control horizon; however, only the first element of the optimal input sequence is applied to the plant.

The plant is modeled as a discrete-time system

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

where \mathcal{U} and \mathcal{X} represent input and state constraints, respectively. The goal of the controller is to drive the system to an equilibrium point by minimizing the cost function (2) while respecting the state and input constraints.

$$J(x, \mathbf{u}) = \sum_{i=0}^{N-1} c(x(i), u(i)) + C(x(k+N)) \quad (2)$$

where $\mathbf{u} = \{u(0), \dots, u(N-1)\}$. Usually, to ensure stability a terminal constraint is imposed.

$$x(k+N) \in X_f \subset \mathcal{X} \quad (3)$$

The optimization problem could be re-written as

$$\mathcal{P}_N(x) : \min_{\mathbf{u}} \{J_N(x, \mathbf{u}) | \mathbf{u} \in \mathcal{U}_N(x)\} \quad (4)$$

where $\mathcal{U}_N(x)$ is the set of feasible control inputs that satisfies the constraints. The control input is given by

$$\kappa_N(x) := u(0; x) \quad (5)$$

Starting from the actual state $x(k)$, the MPC algorithm

- 1) solves the optimal problem $\mathcal{P}_N(x)$ finding the sequence \mathbf{u} ,
- 2) applies the input $u(0; x)$ to the system, and
- 3) repeats for $x(k+1)$.

For a more detailed discussion about MPC and its properties, the reader is referred to [5].

B. Mixed Integer Linear Programming

The mixed integer linear programming (MILP) is used as a tool to incorporate *logical constraints* in the problem formulation as described in [12]. Let $\delta_i \in \{0, 1\}$ be a logical variable such that if $\delta_i = 1(0)$ a statement X_i is true(false) (an example of statement could be $g(x) \geq 0$ or “task accomplished”). Binary operators (e.g., AND, OR, NOT, and implication) can be included in the formulation. For instance, if we want statement X_1 OR statement X_2 to be true then

$$\delta_1 + \delta_2 \geq 1 \quad (6)$$

has to be included in the constraint set.

It is also required to establish relationships between the continuous variables and the logical variables. In order to do so, the well-known **big M** technique is used [16]. The big M technique allows us to convert logical constraints into mixed integer linear inequalities. For instance, let us consider the following implication

$$\delta = 1 \rightarrow f(x) \leq 0 \quad (7)$$

as a constraint. Then it becomes the following mixed integer

linear inequality

$$f(x) \leq M(1 - \delta) \quad (8)$$

where M is an upper bound of $f(x)$. The big M technique is a powerful tool that allows to model systems that include continuous dynamics and logical rules.

Unfortunately, the MILP approach is an *NP-complete* problem, which means that in the worst case, the solution time grows exponentially with the size of the problem. Despite this, algorithms that solve the problem in a reasonable time in practice do exist.

III. PROBLEM FORMULATION

In this work, we consider the robots would navigate in an environment that consists of static obstacles and moving threats. It is assumed that robots are capable of estimating the velocity of the moving threats. Both obstacles and threats are represented by convex linear sets, more formally

$$OX \leq r \quad X = \begin{pmatrix} x \\ y \end{pmatrix} \quad (9)$$

where O is an $R \times 2$ matrix with R the number of linear constraints needed to define an obstacle/threat. A moving threat is described by the same kind of inequality (9) with r being a time dependent vector.

For the sake of simplicity, the problem is formulated with the presence of one static obstacle and one moving threat; however, it could be easily extended to multiple static obstacles and multiple moving threats as shown in the simulation results.

Two different missions are considered:

- 1) *A target assignment problem*: There is a set of known targets that must be reached by the robots. In this case, it is assumed that the number of robots N_R and the number of targets N_T are the same.
- 2) *Go to a region*: In this case, all the robots have to reach a defined region. The target region is *protected* by moving obstacles (i.e., threats) that the robots have to avoid. Again, the target region is described by a set of inequalities as in (9).

It is also assumed that there is a lower level controller in the hierarchy that handles all the nonlinearities in the dynamics of the robots. For this reason, each robot is modeled as a point mass i.e., $\ddot{x} = u$ ¹. The discretized model becomes

$$\mathbf{X}(k+1) = \mathbf{A}\mathbf{X}(k) + \mathbf{B}U(k) \quad (10)$$

with

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & \Delta T & 0 \\ 0 & 1 & 0 & \Delta T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \mathbf{B} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ \Delta T & 0 \\ 0 & \Delta T \end{pmatrix} \quad (11)$$

¹This assumption is quite common in the current multi-robot literature

and

$$\mathbf{X}(k) = \begin{bmatrix} x(k) \\ y(k) \\ v_x(k) \\ v_y(k) \end{bmatrix} \quad \mathbf{U}(k) = \begin{bmatrix} u_x(k) \\ u_y(k) \end{bmatrix} \quad (12)$$

A. On-line Optimization

In this section, we investigate the problem of generating trajectories for a robot formation that reach a target zone while minimizing energy consumption, and avoiding obstacles and inter-vehicle collisions. This can be formulated as an MPC/MILP optimization problem:

$$\min \sum_{k=1}^T \sum_{j=1}^{N_R} [z_{x_j}(k) + z_{y_j}(k)] \quad (13)$$

subject to

$$\begin{aligned} \mathbf{X}_j(k+1) &= A\mathbf{X}_j(k) + B\mathbf{U}_j(k) \\ \forall j &= 1, \dots, N_R \quad \forall k = 0, \dots, T-1. \end{aligned} \quad (14)$$

Equation (14) describes the dynamics of the robots;

$$\begin{aligned} z_{x_j}(k) &\geq u_{x_j}(k) \\ z_{x_j}(k) &\geq -u_{x_j}(k). \end{aligned} \quad (15)$$

The auxiliary variable z_{x_j} in (15) is used to implement the absolute value of u_{x_j} . There are analogous equations for u_{y_j} ;

$$|u_{x,y}| \leq U_{max}, \quad (16)$$

represents the input bounds;

$$o_{p1}x_j(k) + o_{p2}y_j(k) \leq r_p \Rightarrow \omega_{pj}^k = 1 \quad (17)$$

$$\forall p = 1, \dots, R; \forall j = 1, \dots, N_R; \forall k = 1, \dots, T;$$

$$\begin{aligned} \sum_{p=1}^R \omega_{pj}^k &\leq R-1 \\ \forall j &= 1, \dots, N_R; \forall k = 1, \dots, T; \end{aligned} \quad (18)$$

are the equations used for obstacle and threat avoidance. Let us consider the j -th robot at the k -th sample time. Equation (17) drives the binary auxiliary variable ω_{pj}^k to 1 if the p -th constraint that describes the obstacle is satisfied by the position coordinates of the robot j at time k . The robot is inside the obstacle if all the R constraints that describe the obstacle are satisfied. The constraint in (18) imposes that the coordinates of the robot violate at least one of the constraints that describe the obstacle. This ensures that the robot stays outside the obstacle.

The collision avoidance constraint can be implemented in a similar way. Specifically, a square safety zone around a robot is defined. Other robots will see this safety zone as a moving obstacle that has to be avoided. The set of equations that

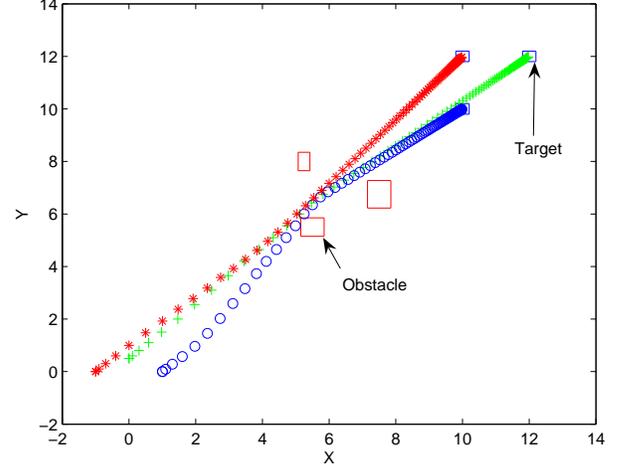


Fig. 1: Three robots moving towards given targets while avoiding static obstacles.

implement collision avoidance are:

$$\begin{aligned} C_A \begin{pmatrix} x_i(k) - x_j(k) \\ y_i(k) - y_j(k) \end{pmatrix} &\leq s_d \\ \sum_{p=1}^4 \tau_{pij}(k) &\leq 3 \end{aligned} \quad (19)$$

$$\forall i = 1, \dots, N_R; j = i+1, \dots, N_R; k = 1, \dots, T$$

where the 2×4 matrix C_A and the vector s_d define a *safety zone* around the robot. τ_{pij} is an auxiliary binary variable which is 1, if the p -th inequality in the first equation in (19) is satisfied.

The following set of equations implements the target assignment constraint:

$$\mathbf{X}_j(T) - \mathbf{X}_{T_l} = 0 \Leftrightarrow \tau_{jl} = 1 \quad (20)$$

$$\forall j = 1, \dots, N_R; l = 1, \dots, N_T$$

$$\sum_{j=1}^{N_R} \tau_{jl} = 1 \quad l = 1, \dots, N_T \quad (21)$$

$$\sum_{l=1}^{N_T} \tau_{jl} = 1 \quad j = 1, \dots, N_R \quad (22)$$

Equation (20) will set the auxiliary binary variable τ_{jl} to 1, if robot j reaches target l at the end of the prediction horizon. Additionally, equation (21) ensures that each target is assigned to only one robot, while equation (22) insures that each robot is assigned to only one target.

$$O_T \begin{pmatrix} x_j(T) \\ y_j(T) \end{pmatrix} \leq r_T \quad (23)$$

Notice that equation (23) implements an alternative assignment behavior. In this case, robots are not required to reach a specific target. Rather the robots' objective is to reach a *zone*

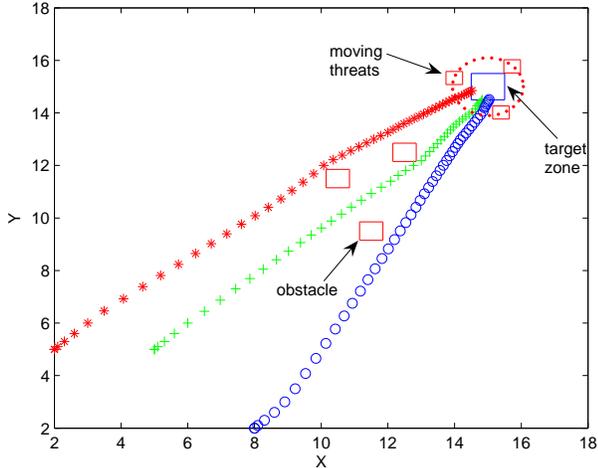


Fig. 2: Three robots moving towards a target region while avoiding obstacles and moving threats.

in the environment. Hence, O_T and r_T define a *target zone*. Finally, all the constraints are transformed into mixed integer linear inequalities using the big M technique.

IV. CASE STUDIES

The formulation presented in the previous section was implemented using Matlab. We used ILOG's CPLEX 9.0 to solve the MILP. The interface available in [17] was used to run CPLEX from Matlab. First, we consider the case in which three robots have to reach three targets while avoiding obstacles and minimizing the energy. The simulation results are depicted in Figure 1. Notice that there is not a pre-assignment of targets. The assignment is determined *on-the-fly* by the MPC/MILP algorithm. Since the problem is solved using a discrete-time model of the system, continuous trajectories may pass through obstacle corners. This problem is easily eliminated by slightly enlarging the obstacles. This is a common practice in planning methods in order to account for the physical dimensions of the robots.

Now, let us assume that a team of robots have to reach a protected target zone (*c.f.*, RoboFlag in [13]). As before, robots have to navigate towards the target zone while minimizing the energy and avoiding collisions. In addition, robots should avoid moving threats that are protecting the target region. This scenario was motivated by [18]. The results obtained running the second simulation are shown in Figure 2. As it can be seen in Figure 3, robots are able to avoid the moving threats and successfully reach the target zone.

A possible extension of the first case is to have moving targets instead of static targets. This kind of scenario would be suitable to implement formation control. In formation control, robots are to navigate maintaining a formation shape. Suppose there are reference trajectories that pass through obstacles. It is possible to use the above target assignment framework such that the robots follow moving targets while minimizing

the energy and avoiding collisions. However, the MPC/MILP has to be modified because a feasible solution will not exist when targets are inside obstacles. To overcome this problem, the target assignment constraint is converted from a *hard* constraint into a *soft* constraint. Thus, when a target is inside an obstacle, no robot is assigned to that target. To transform the target assignment constraint from a hard constraint to a soft constraint, equations (21) and (22) are modified as follows:

$$\begin{aligned} \sum_{j=1}^{N_R} \tau_{jl} + \zeta_l &= 1 \quad l = 1, \dots, N_T \\ \sum_{l=1}^{N_T} \tau_{jl} + \xi_j &= 1 \quad j = 1, \dots, N_R \end{aligned} \quad (24)$$

where ζ and ξ are auxiliary continuous variables. The original constraint is violated if the auxiliary variable is not zero. The cost function has to be modified to add a penalty in case the constraint is violated. Thus, the new cost function becomes

$$\min \sum_{k=1}^T \sum_{j=1}^{N_R} [z_{x_j}(k) + z_{y_j}(k)] + W_1 \sum_{i=1}^{N_T} \zeta_i + W_2 \sum_{j=1}^{N_R} \xi_j \quad (25)$$

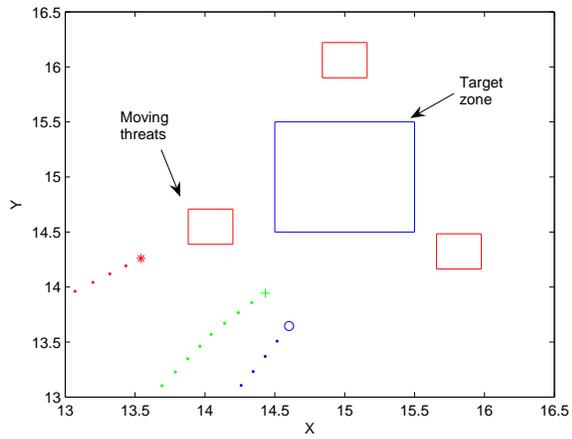
where W_1 and W_2 are positive large numbers that penalize the violation of the constraint. The result of the simulation of this case study is reported in Figure 4(a) and Figure 4(b).

Figure 4(a) depicts robots starting from some initial conditions and following a set of moving targets. Time-varying targets might represent a desired formation shape. Figure 4(b), on the other hand, shows robots starting from the same initial conditions but with an obstacle in the path of a target. Notice that the algorithm executes target assignment *on-the-fly*.

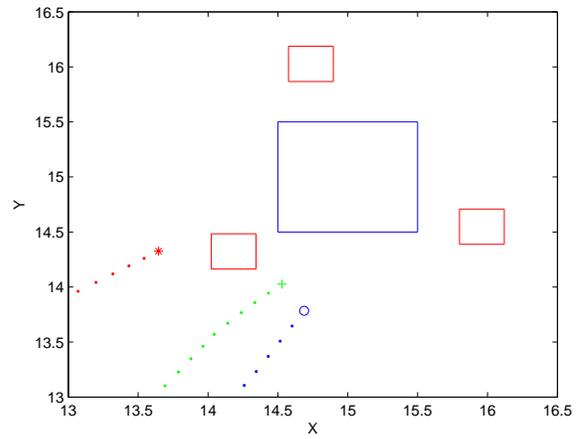
V. COMPLEXITY ANALYSIS

The solution of the MILP is based on the branch and bound algorithm. The basic idea of the branch and bound algorithm is to relax the binary constraints and solve the equivalent linear problem. If the solution of the linear problem is integer feasible, then this solution is also the solution of the MILP. If the solution is not integer feasible, one of the binary variables δ_i that is not integer is taken and two new problems are formulated: (1) adding the constraint $\delta_i = 0$, and (2) with the constraint $\delta_i = 1$. This procedure generates a binary tree where each node corresponds to a formulation of a particular linear problem. Once an integer feasible solution is found, the bound phase helps in speeding up the search of the optimal solution. For a more detailed overview of the branch and bound algorithm see [16]. It should be clear from the previous discussion that in the worst case, the number of linear problems that have to be solved grow exponentially with the number of binary variables present in the MILP.

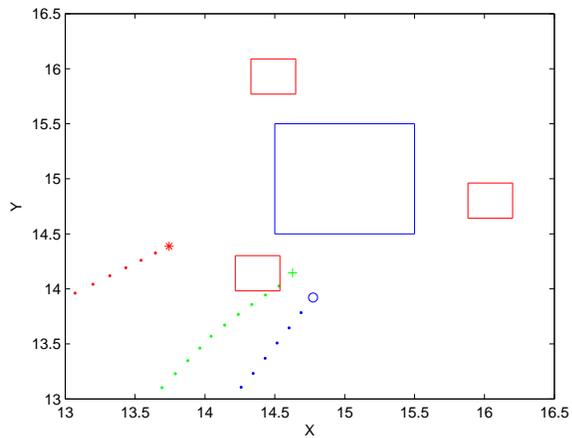
An analysis of the number of binary variables needed to implement each constraint can help in predicting the difficulty of the problem. In the above formulation constraints are defined by: the equations of motion of the robots (14); the limits on the inputs (16), the implementation of the absolute value of the input (15), and the *go-to-region* behavior (23).



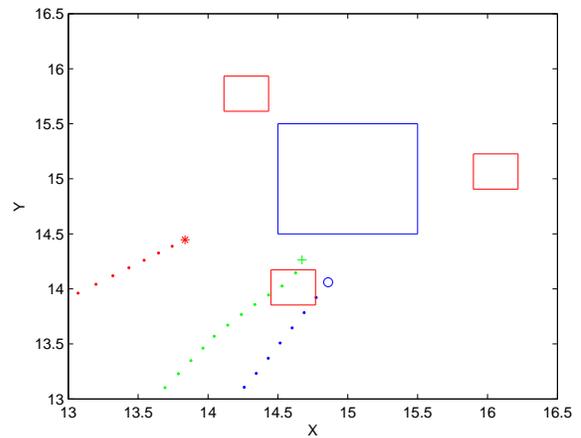
(a) $k = 48$



(b) $k = 49$

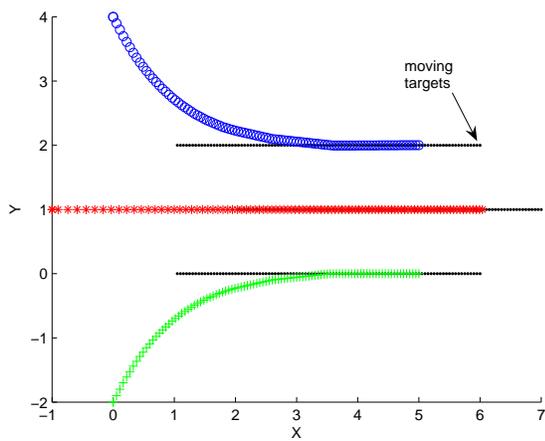


(c) $k = 50$

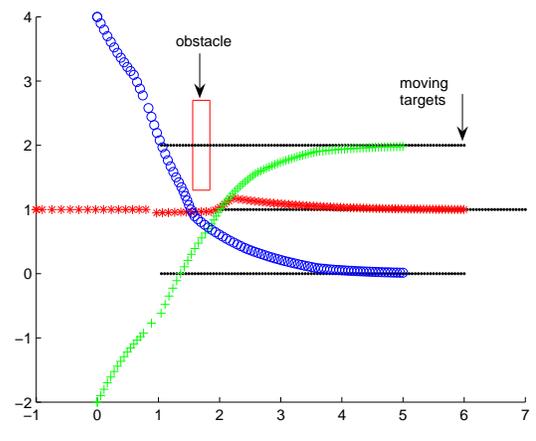


(d) $k = 51$

Fig. 3: Robots are able to avoid moving obstacles and reach the goal zone.



(a) Moving targets.



(b) Moving targets with obstacles and dynamic reassignment.

Fig. 4: Formation control using moving targets.

All these constraints do not have binary variables, and so do not affect much the complexity of the problem. However, other constraints such as obstacle avoidance (17)- (18), collision avoidance (19), and target assignment (20), (21), and (22), contain binary variables. These last group of constraints strongly affect the time required for achieving a solution of the problem. In particular, for the obstacle avoidance if R_i is the number of equations required to define the obstacle and N_o is the number of obstacles, the number of binary variables needed is $N_R T \sum_{i=1}^{N_o} R_i$, where N_R and T are the number of robots and the control horizon, respectively.

Furthermore, to implement the target assignment behavior, seven binary variables for each possible pair $\{robot, target\}$ are necessary. The total number of binary variables for target assignment is $7N_R N_T$. Finally, for collision avoidance $4T((N_R - 1)^2 - \sum_{j=1}^{N_R-2} j)$ are required. Suppose a given mission consists of 5 robots, 3 rectangular obstacles, collision avoidance, go-to-region type of task, and a control horizon of 20 steps, then 2000 binary variables are required. For this configuration, each optimization problem is solved with an average time of 4.37 s. It has been noted that the time required for obtaining a solution is also related to the instance of the problem. For example, for the same configuration with 7 robots starting from some initial conditions 89 s. are needed, while from some other initial conditions, the time required can grow up to 1065 s. This is due to the bounding phase of the branch and bound algorithm that in some cases is much more efficient than in others. Finding a way to initialize the tree search such that the bounding phase is accelerated is in our research agenda.

VI. CONCLUSIONS

In this paper, we investigate on-line optimization-based strategies to coordinate a team of robots. The robots are engaged in target assignment missions within a dynamic environment. Several case studies are used to demonstrate the feasibility of combining MPC and MILP to solve a class of multi-vehicle coordination problems.

In our future work, we plan to investigate reformulating the MILP portion of our control problem as a quadratically constrained quadratic program (QCQP) and solving through convex approximation techniques. An inherent problem with modeling obstacles as exclusion regions is the creation of voids in the feasible set. As a consequence, the resulting problem formulation will not be convex. Most of the literature on solving QCQPs centers on the special case where the constraint functions are convex. However, there are convex relaxations of the original optimization problem that provide useful bounds on the feasible solutions [19], [20]. The cost of this relaxation is that the optimal value returned for the relaxed formulation may differ from the true optimum by an arbitrary duality gap. However, the MILP solution itself provides no guarantees regarding optimality. An advantage of employing convex relaxations is that these can be formulated as semidefinite programs (SDP), and we can take advantage of the efficient solution schemes developed for this class of

problems. We plan to compare the two approaches in terms of performance and computational efficiency in the near future.

ACKNOWLEDGMENTS

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

REFERENCES

- [1] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Trans. on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, April 2004.
- [2] J. T. Feddema, R. D. Robinett, and R. H. Byrne, "An optimization approach to distributed controls of multiple robot vehicles," in *Workshop on Control and Cooperation of Intelligent Miniature Robots, IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, Nevada, October 31 2003.
- [3] 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.
- [4] S. Zelinski, T. J. Koo, and S. Sastry, "Optimization-based formation reconfiguration planning for autonomous vehicles," in *Proc. IEEE Int. Conf. Robot. Automat.*, Taipei, Taiwan, September 2003, pp. 3758–3763.
- [5] D. Q. Mayne, J. B. Rawings, C. V. Rao, and P. O. M. Sokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, June 2000.
- [6] A. Jadbabaie, J. Yu, and J. Hauser, "Unconstrained receding-horizon control of nonlinear systems," *IEEE Trans. on Automatic Control*, vol. 46, no. 5, pp. 776–783, May 2001.
- [7] J. Bullingham, A. Richards, and J. P. How, "Receding horizon control of autonomous aerial vehicles," in *Proc. American Control Conference*, Anchorage, AK, May 2002, pp. 3741–3746.
- [8] R. Fierro and K. Wesselowski, "Optimization-based control of multi-vehicle systems," in *A Post-Workshop Volume 2003 Block Island Workshop on Cooperative Control Series*, ser. LNCIS, V. Kumar, N. Leonard, and A. Morse, Eds. Springer, 2004, vol. 309, pp. 63–78.
- [9] W. B. Dunbar and R. M. Murray, "Distributed receding horizon control with application to multi-vehicle formation stabilization," *Automatica*, August 2004. Preprint.
- [10] T. Keviczky, F. Borrelli, and G. J. Balas, "A study on decentralized receding horizon control for decoupled systems," in *Proc. American Control Conference*, Boston MA, June 2004.
- [11] K. Wesselowski and R. Fierro, "A dual-mode model predictive controller for robot formations," in *Proc. IEEE Conf. on Decision and Control*, Maui, Hawaii, Dec. 2003, pp. 3615–3620.
- [12] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.
- [13] M. G. Earl and R. D'Andrea, "Modeling and control of a multi-agent system using mixed integer linear programming," in *Proc. IEEE Conf. on Decision and Control*, vol. 1, Las Vegas, NV, Dec 10-13 2002, pp. 107–111.
- [14] A. Richards, Y. Kuwata, and J. How, "Experimental demonstrations of real-time MILP control," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*. Austin, Texas: AIAA, August 2003, pp. 1–11.
- [15] ILOG, *CPLEX 9.0 User's Manual*, 2004.
- [16] H. Williams, *Model Building in Mathematical Programming*. New York: John Wiley & Sons, 1985.
- [17] M. Baotic, *Matlab interface to CPLEX*. Available from <http://control.ee.ethz.ch/hybrid/cplexint.php>.
- [18] J. S. Baras, X. Tan, and P. Holdership, "Decentralized control of autonomous vehicles," in *Proc. IEEE Conf. on Decision and Control*, Maui, Hawaii, Dec. 2003, pp. 1532–1537.
- [19] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, UK: Cambridge University Press, March 2004.
- [20] —, *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.