2005 American Control Conference
June 8-10, 2005. Portland, OR, USA

Linear-Programming-Based Multi-Vehicle Path Planning with Adversaries

Georgios C. Chasparis and Jeff S. Shamma
Department of Mechanical and Aerospace Engineering
University of California Los Angeles
Box 951597, Los Angeles, CA 90095
{gchas,shamma}@seas.ucla.edu

Abstract— **A linear-programming (LP) based path planning algorithm is developed for deriving optimal paths for a group of autonomous vehicles in an adversarial environment. In this method, both friendly and enemy vehicles are modelled as different resource types in an arena of sectors, and the path planning problem is viewed as a resource allocation problem. Simple model simplifications are introduced to allow the use of linear programming in conjunction with a receding horizon implementation for multi-vehicle path planning. Stochastic models based on the current position of opposing vehicles are used to describe their possible future trajectories. The utility of the LP-based algorithm is tested in the RoboFlag drill, where both teams of vehicles have equal path planning capabilities using the proposed algorithm. Results show that the LP-based path planning in combination with a simple enemy model can be used for efficient multi-vehicle path planning in an adversarial environment.**

I. Introduction

One problem in autonomous multi-vehicle systems is the real-time derivation of vehicle paths. Often this problem can be formulated as a large-scale optimization. However, environmental conditions are not necessarily stationary, and the inclusion of these uncertainties to the optimization problem is an open issue.

Several optimization methods already have been tested for multi-vehicle path planning. One is based on the notion of *coordination variables* [1], where trajectories are determined so that threat avoidance is ensured and timing constraints are satisfied. However, the location of the considered threats are deterministically known.

Several papers on mission planning of UAVs construct *Voronoi-based polygonal paths* from the currently known location of the threats. Among those paths the lowest-cost flyable path can be computed [2]. In [3] and [4] a probabilistic approach is introduced, where a probability of a threat or target is assumed to be known. According to [3], global strategies may be computationally inefficient, while the path generated by the strategy might get in a limit cycle.

Since several classes of multi-vehicle systems can be modelled as hybrid systems, one of the suggested approaches to designing feedback controllers is based on *model predictive control* [5], where an optimization problem is solved based on a prediction of the future evolution of the system.

For some classes of multi-vehicle systems, this optimization is a *mixed integer linear programming* problem [6], [7]. However, the computation time can be very large, while probabilities of the threats cannot easily be included in the optimization. *Dynamic programming* can take into account such probabilities, however, their calculation is computationally impractical [8].

In this paper, we seek a linear formulation of the problem. The potential advantage is that a linear program is computationally appealing. The proposed approach is based on a linear model for resource allocation in an adversarial environment [9]. Both friendly and enemy vehicles are modelled as different resource types in an arena of sectors, and the path planning problem is viewed as a resource allocation problem. However, the resulting linear dynamic model is subject to binary optimization constraints, while the enemy's future locations are unknown.

Model simplifications are introduced to allow the use of linear programming in conjunction with a receding horizon implementation for multi-vehicle path planning. Stochastic models based on the current position of opposing vehicles are used to describe their possible future trajectories. The utility of the LP-based algorithm is tested in the RoboFlag drill, where both teams of vehicles have equal path planning capabilities using the proposed algorithm.

II. Problem Formulation

A. State-space model

We consider a model that describes the movement and engagement of friendly and enemy resources in an arena of sectors [9]. The battlefield is divided into a collection of sectors, S, and evolution is in discrete time.

A vehicle is represented as a resource type, r_j. We define its quantity level at sector s_i to be $x_{s_i,r_j} \in B \triangleq \{0,1\}$, so that resource level "1" corresponds to the sector where the vehicle lies in, otherwise it is "0". Under these assumptions, the state of each resource type, r_j, is

$$\mathbf{x}_{r_j} = \begin{pmatrix} x_{s_1,r_j} & x_{s_2,r_j} & \dots & x_{s_{n_s},r_j} \end{pmatrix}^T \in B^{n_s}$$

This work was supported by AFOSR/MURI grant #F49620-01-1-0361.

where n_s is the total number of sectors in S. Thus, the state of a collection, R, of n_r vehicles could be:

$$\mathbf{x} = \left(\begin{array}{cccc} \mathbf{x}_{r_1}^T & \mathbf{x}_{r_2}^T & \dots & \mathbf{x}_{r_{n_r}}^T \end{array} \right)^T \in B^{n_x}$$

where $n_x = n_s n_r$ is the total number of states.

Resource level changes represent vehicles movement. They can either remain in the same sector or move to a neighboring sector. Movements within a sector are not modelled. Therefore, the control action includes the transitions of each resource type $r_j \in R$ from sector $s_i \in S$ to a neighboring sector $s_k \in N(s_i, r_j)$, where $N(s_i, r_j)$ is the set of neighboring sectors of sector s_i that can be reached by resource type r_j in one time-stage.

Define $u_{s_i \leftarrow s_k, r_j} \in B$ as the level of resource type r_j that is being transferred from sector s_k to sector s_i, where $s_k \in N(s_i, r_j)$. Then the system evolves according to the following state-space equations:

$$x_{s_i,r_j}^+ = x_{s_i,r_j} + \sum_{s_k \in N(s_i,r_j)} u_{s_i \leftarrow s_k, r_j} - \sum_{s_k \in N(s_i,r_j)} u_{s_k \leftarrow s_i, r_j} \quad (1)$$

for each $s_i \in S$ and $r_j \in R$, where superscript "+" denotes the next time-stage. In order for this set of equations to describe a continuous flow of resources, the following constraint must also be satisfied:

$$0 \leq \sum_{s_k \in N(s_i,r_j)} u_{s_k \leftarrow s_i, r_j} \leq x_{s_i,r_j}, \quad \forall s_i \in S, \quad \forall r_j \in R. \quad (2)$$

Define the control vector $\mathbf{u}_{s_i,r_j} \in B^{(n_s-1)}$ as the resource levels of type $r_j \in R$ that enter $s_i \in S$, i.e.,

$$\mathbf{u}_{s_i,r_j} = \left(\begin{array}{cc} u_{s_i \leftarrow s_1, r_j} \dots u_{s_i \leftarrow s_{i-1}, r_j} & u_{s_i \leftarrow s_{i+1}, r_j} \dots u_{s_i \leftarrow s_{n_s}, r_j} \end{array} \right)^T$$

The control vector $\mathbf{u}_{r_j} \in B^{n_s(n_s-1)}$ consists of all transitions of $r_j \in R$, i.e.,

$$\mathbf{u}_{r_j} = \left(\begin{array}{cccc} \mathbf{u}_{s_1,r_j}^T & \mathbf{u}_{s_2,r_j}^T & \dots & \mathbf{u}_{s_{n_s},r_j}^T \end{array} \right)^T.$$

Then, the control vector of the collection of resource types or vehicles, R, is

$$\mathbf{u} = \left(\begin{array}{cccc} \mathbf{u}_{r_1}^T & \mathbf{u}_{r_2}^T & \dots & \mathbf{u}_{r_{n_r}}^T \end{array} \right)^T \in B^{n_u}$$

where $n_u = n_s n_r (n_s - 1)$ is the total number of controls.

It is not difficult to show that Equations (1) and (2), which describe the system's evolution, can take on the following form:

$$\left\{ \begin{array}{l} \mathbf{x}^+ = \mathbf{x} + \mathbf{B}_{\text{in}} \cdot \mathbf{u} - \mathbf{B}_{\text{out}} \cdot \mathbf{u} = \mathbf{x} + \mathbf{B} \cdot \mathbf{u} \\ \mathbf{0} \leq \mathbf{B}_{\text{out}} \cdot \mathbf{u} \leq \mathbf{x} \\ \mathbf{x} \in B^{n_x}, \quad \mathbf{u} \in B^{n_u} \end{array} \right. \quad (3)$$

where $\mathbf{B} = \mathbf{B}_{\text{in}} - \mathbf{B}_{\text{out}}$. Note that the entries of both $\mathbf{x}$ and $\mathbf{u}$ take on only binary values.

For example, in case of two sectors and one vehicle, $S = \{s_1, s_2\}$, $R = \{r_1\}$, $N(s_1, r_1) = \{s_2\}$, $N(s_2, r_1) = \{s_1\}$, and the state-space equations are:

$$\underbrace{\left(\begin{array}{c} x_{s_1,r_1} \\ x_{s_2,r_1} \end{array} \right)^+}_{\mathbf{x}^+} = \underbrace{\left(\begin{array}{c} x_{s_1,r_1} \\ x_{s_2,r_1} \end{array} \right)}_{\mathbf{x}} + \underbrace{\left[\begin{array}{cc} 1 & -1 \\ -1 & 1 \end{array} \right]}_{\mathbf{B}} \cdot \underbrace{\left(\begin{array}{c} u_{s_1 \leftarrow s_2, r_1} \\ u_{s_2 \leftarrow s_1, r_1} \end{array} \right)}_{\mathbf{u}}.$$

We can also define:

$$\mathbf{B}_{\text{in}} = \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right], \quad \mathbf{B}_{\text{out}} = \left[\begin{array}{cc} 0 & 1 \\ 1 & 0 \end{array} \right].$$

B. Single Resource Models

An alternative model is to view all vehicles as a single resource type. In this case, the state of the system can be defined as

$$\mathbf{x}_1 = \mathbf{x}_{r_1} + \mathbf{x}_{r_2} + \ldots + \mathbf{x}_{r_{n_r}} \in B^{n_{x,1}}$$

where $n_{x,1} = n_s$. Similarly, we define

$$\mathbf{u}_1 = \mathbf{u}_{r_1} + \mathbf{u}_{r_2} + \ldots + \mathbf{u}_{r_{n_r}} \in B^{n_{u,1}}$$

where $n_{u,1} = n_s(n_s - 1)$. Thus, for suitable matrices $\mathbf{B}_1$ and $\mathbf{B}_{\text{out},1}$, we can write

$$\left\{ \begin{array}{l} \mathbf{x}_1^+ = \mathbf{x}_1 + \mathbf{B}_1 \cdot \mathbf{u}_1 \\ \mathbf{0} \leq \mathbf{B}_{\text{out},1} \cdot \mathbf{u}_1 \leq \mathbf{x}_1 \\ \mathbf{x}_1 \in B^{n_{x,1}}, \quad \mathbf{u}_1 \in B^{n_{u,1}} \end{array} \right. \quad (4)$$

This state-space representation has fewer states and controls than the one of (3).

C. Adversarial environment

We consider the case that an adversarial team of vehicles also evolves within the same arena of sectors. The two opposing teams are subject to attrition, where attrition could be interpreted as the result of collisions of opponent vehicles. In this case, a possible objective of each team is to cause the largest possible attrition to their enemies.

We model enemy vehicles to follow similar state-space equations as those of friendly vehicles. We also assume that the decisions of both teams are made at the same time instants and that the current state of the opposing resources is known.

Let superscript "f" denote the friendly team, and superscript "e" denote the enemy team. Then the evolution of both friendly and enemy vehicles can be described by

$$\left\{ \begin{array}{l} \left(\mathbf{x}^i\right)^+ = \mathbf{x}^i + \mathbf{B}^i \cdot \mathbf{u}^i - \mathbf{d}^i(\mathbf{x}^i, \mathbf{x}^{-i}) \\ \mathbf{0} \leq \mathbf{B}_{\text{out}}^i \cdot \mathbf{u}^i \leq \mathbf{x}^i \\ \mathbf{x}^i \in B^{n_x^i}, \quad \mathbf{u}^i \in B^{n_u^i} \end{array} \right. , \quad i \in \{f, e\} \quad (5)$$

where $\mathbf{d}^i$ is the attrition function that depends on the current state of each team and $-i$ is the opposite of i.

D. Model simplifications

The state-space equations of (5) cannot be used to formulate a linear optimization program for future friendly planning. The main obstacles are:

- The presence of the attrition function, which is generally nonlinear.
- The unknown control vector of the enemy resources.

In [9], a similar state-space model, but without the binary constraints, is approximated with a linear model based on two model simplifications that will allow the use of linear programming.

First, we remove the attrition function from the state-space equations of (5), i.e.,

$$\left\{ \begin{array}{l} \left(\mathbf{x}^i\right)^+ = \mathbf{x}^i + \mathbf{B}^i \cdot \mathbf{u}^i \\ \mathbf{0} \leq \mathbf{B}^i_{\text{out}} \cdot \mathbf{u}^i \leq \mathbf{x}^i \\ \mathbf{x}^i \in B^{n^i_x}, \quad \mathbf{u}^i \in B^{n^i_u} \end{array} \right. , \quad i \in \{f, e\}. \tag{6}$$

In other words, we assume that both teams evolve as if no attrition will occur.

Second, since the controls of the enemy team are not known to the friendly team, we assume that the enemy team implements an *assumed* feedback policy $\mathbf{G}^e \in \bar{B}^{n^e_u \times n^e_x}$, such that

$$\mathbf{u}^e = \mathbf{G}^e \cdot \mathbf{x}^e \tag{7}$$

where $\bar{B} \triangleq [0,1]$.

Due to these two model simplifications, we can expect that the resulting model of (6) and (7) will be significantly different from the actual evolution described by (5). We overcome this problem by applying a receding horizon strategy [5].

E. Enemy Modelling

The enemy's feedback matrix, $\mathbf{G}^e$, contains the *assumed* information about the future states of the enemy resources. It is generally unknown to the friendly resources but introduced for the sake of prediction in optimization. This feedback matrix can be used for modelling several behaviors, such as

- anticipated paths of enemy resources,
- diffusion of enemy resources,
- probability maps of enemy resources.

In particular, for any $s_i \in S$, $r_j \in R^e$ and $s_k \in N^e(s_i, r_j)$, we assume that

$$u^e_{s_k \leftarrow s_i, r_j} = g^e_{s_k \leftarrow s_i, r_j} \cdot x^e_{s_i, r_j} \tag{8}$$

where $g^e_{s_k \leftarrow s_i, r_j}$ is the assumed feedback of the enemy resource type r_j.

Setting $g^e_{s_k \leftarrow s_i, r_j} \in \{0,1\}$, we define an *anticipated* next destination of the opposing resource type r_j. If we split the resource level $x^e_{s_i, r_j}$ to two or more destination sectors, i.e., $g^e_{s_k \leftarrow s_i, r_j} < 1$, then we create a *diffusion* of enemy resources. Finally, $g^e_{s_k \leftarrow s_i, r_j}$ can be interpreted as the *probability* that opposing resource type r_j will move from sector s_i to sector s_k.

In either case, the following properties must be satisfied

$$\left\{ \begin{array}{l} g^e_{s_k \leftarrow s_i, r_j} \in [0,1] \\ \sum_{s_k \in N(s_i, r_j)} \{g^e_{s_k \leftarrow s_i, r_j}\} \leq 1 \end{array} \right. \tag{9}$$

which guarantee that the control constraints of (2) hold.

In case of two sectors and one opposing vehicle, we have:

$$\underbrace{\begin{pmatrix} u^e_{s_1 \leftarrow s_2, r_1} \\ u^e_{s_2 \leftarrow s_1, r_1} \end{pmatrix}}_{\mathbf{u}^e} = \underbrace{\begin{bmatrix} 0 & g^e_{s_1 \leftarrow s_2, r_1} \\ g^e_{s_2 \leftarrow s_1, r_1} & 0 \end{bmatrix}}_{\mathbf{G}^e} \cdot \underbrace{\begin{pmatrix} x_{s_1, r_1} \\ x_{s_2, r_1} \end{pmatrix}}_{\mathbf{x}^e}.$$

For example, if $g^e_{s_1 \leftarrow s_2, r_1} = 0.3$, then 30% of x_{s_2, r_1} will move from sector s_2 to sector s_1, while the rest of it will remain in sector s_2.

The great advantage of the introduction of the enemy's feedback matrix is that we can model enemy's intentions based on their current location or even their velocity.

III. Optimization Set-up

A. Objective function

The simplified system of friendly and enemy vehicles is described by a system of linear equations and constraints, (6) and (7). We now introduce a linear objective function that will allow the use of linear programming in deriving optimal friendly paths. Optimal paths are described by a sequence of states.

For each team $i \in \{f, e\}$, define the vector of optimized states for a finite optimization horizon, T_p, as

$$\mathbf{X}^i = \begin{pmatrix} \mathbf{x}^i[1]^T & \mathbf{x}^i[2]^T & \ldots & \mathbf{x}^i[T_p]^T \end{pmatrix}^T$$

where $\mathbf{x}^i[t] \in B^{n^i_x}$ is the state vector at the t^{th} future time-stage. We can also define the vector $\mathbf{X}^i_1$, where all vehicles of the collection R^i are considered as a single resource type, i.e.,

$$\mathbf{X}^i_1 = \begin{pmatrix} \mathbf{x}^i_1[1]^T & \mathbf{x}^i_1[2]^T & \ldots & \mathbf{x}^i_1[T_p]^T \end{pmatrix}^T.$$

Possible objectives in an adversarial environment are:

- Minimization of intercepted friendly vehicles. (evasion)
- Maximization of intercepted enemy vehicles. (pursuit)
- Tracking of a reference state vector. (surveillance)

These constraints can be represented by a linear objective function of the form:

$$\min_{\mathbf{X}^f_1, \mathbf{U}^f_1} \left[\alpha^f \cdot \mathbf{X}^e_1 + \beta^f \cdot \mathbf{X}^f_{\text{ref}}\right]^T \cdot \mathbf{X}^f_1. \tag{10}$$

The inner product of the friendly vector of optimized states, $\mathbf{X}^f_1$, with the corresponding enemy vector, $\mathbf{X}^e_1$, increases with the number of interceptions between friendly and enemy vehicles. Therefore, in case $\alpha^f < 0$, interceptions of enemy vehicles are encouraged, while $\alpha^f > 0$ causes friendly vehicles to avoid enemy vehicles. Moreover, $\beta^f < 0$ encourages the friendly states to become aligned to the reference ones, $\mathbf{X}^f_{\text{ref}}$. We can always take

$$\left|\alpha^f\right| + \left|\beta^f\right| = 1.$$

B. Constraints

The objective function of (10) is subject to the dynamics of (6) and (7), throughout the optimization horizon T_p. In particular, the following equations must be satisfied for each $t \in T \triangleq \{0, 1, \ldots, T_p - 1\}$:

$$\mathbf{x}^f_1[t+1] \quad = \quad \mathbf{x}^f_1[t] + \mathbf{B}^f_1 \cdot \mathbf{u}^f_1[t]. \tag{11}$$

Define

$$\mathbf{U}^f_1 = \begin{pmatrix} \mathbf{u}^f_1[0]^T & \mathbf{u}^f_1[1]^T & \ldots & \mathbf{u}^f_1[T_p - 1]^T \end{pmatrix}^T.$$

There exist matrices $\mathbf{T}^f_{\mathrm{xx}_0}$ and $\mathbf{T}^f_{\mathrm{xu}}$ such that the dynamic equations of (11) are written equivalently as

$$\mathbf{X}^f_1 = \mathbf{T}^f_{\mathrm{xx}_0} \cdot \mathbf{x}^f_1[0] + \mathbf{T}^f_{\mathrm{xu}} \cdot \mathbf{U}^f_1. \tag{12}$$

The control vector $\mathbf{u}^f_1[t]$ for each $t \in T$ must also satisfy:

$$\mathbf{B}^f_{\mathrm{out},1} \cdot \mathbf{u}^f_1[t] \leq \mathbf{x}^f_1[t]. \tag{13}$$

It is straightforward to construct matrices $\mathbf{T}^f_{\mathrm{xu,c}}$ and $\mathbf{T}^f_{\mathrm{xx_0,c}}$, such that the constraints of (13) take on the following form:

$$\mathbf{T}^f_{\mathrm{xu,c}} \cdot \mathbf{U}^f_1 \leq \mathbf{T}^f_{\mathrm{xx_0,c}} \cdot \mathbf{x}^f[0]. \tag{14}$$

Furthermore, obstacle avoidance easily can be represented by orthogonality constraints, such as

$$(\mathbf{X}^f_{\mathrm{obs}})^T \cdot \mathbf{X}^f_1 = 0 \tag{15}$$

where $\mathbf{X}^f_{\mathrm{obs}} \in B^{T_p n_s}$ is a vector whose entries are equal to "1" if they correspond to the obstacles' locations, and "0" otherwise.

C. Mixed-integer linear optimization

The objective function of (10) in conjunction with the dynamic constraints of (12), the control constraints of (14) and the obstacle avoidance constraints of (15), formulate a *mixed integer linear programming* optimization for friendly planning. This optimization problem can be written as:

$$\begin{array}{ll} \text{minimize} & \left[\alpha^f \cdot \mathbf{X}^e_1 + \beta^f \cdot \mathbf{X}^f_{\mathrm{ref}}\right]^T \cdot \mathbf{X}^f_1 \\ \text{subject to} & \mathbf{T}^f_{\mathrm{xu,c}} \cdot \mathbf{U}^f_1 \leq \mathbf{T}_{\mathrm{xx_0,c}} \cdot \mathbf{x}^f_1[0] \\ & \mathbf{X}^f_1 - \mathbf{T}^f_{\mathrm{xu}} \cdot \mathbf{U}^f_1 = \mathbf{T}^f_{\mathrm{xx}_0} \cdot \mathbf{x}^f_1[0] \\ & (\mathbf{X}^f_{\mathrm{obs}})^T \cdot \mathbf{X}^f_1 = 0 \\ \text{variables} & \mathbf{X}^f_1 \in B^{T_p n^f_{x,1}}, \quad \mathbf{U}^f_1 \in B^{T_p n^f_{u,1}}. \end{array} \tag{16}$$

The vector of states of enemy resources, $\mathbf{X}^e_1$, is not known. As previously discussed, we can assume that enemy resources evolve according to an *assumed* feedback law, $\mathbf{G}^e$, such that for each $t \in T$,

$$\mathbf{x}^e[t+1] = (\mathbf{I} + \mathbf{B}^e\mathbf{G}^e)^{t+1} \cdot \mathbf{x}^e[0]. \tag{17}$$

Hence, there exists matrix $\mathbf{T}^e_{\mathrm{xx_0,G}}$ such that the state-space equations of (17) are written as

$$\mathbf{X}^e_1 = \mathbf{T}^e_{\mathrm{xx_0,G}} \cdot \mathbf{x}^e_1[0]. \tag{18}$$

IV. LP-BASED PATH PLANNING

The optimization of (16) is a mixed-integer linear programming problem, where both states and controls take values on $B = \{0, 1\}$. Although there are several methods that can be used for computing the optimal solution, such as *cutting plane methods* or *branch and bound* [10], we prefer to solve a linear programming problem instead. The main reason for this preference is the computational complexity of an integer problem.

To this end, we transform the mixed integer program of (16) into a *linear-programming-based optimization planning*. This planning includes the following steps:

1) We introduce the *linear programming relaxation* of the *mixed integer programming problem* of (16).
2) Given the non-integer optimal solution of the *linear programming relaxation*, we compute a suboptimal solution of the *mixed integer programming problem.*
3) We apply this solution according to a *receding horizon implementation.*

A. Linear programming relaxation

The linear programming relaxation of (16) assumes that the vector of optimized states, $\mathbf{X}^f_1$, and controls, $\mathbf{U}^f_1$, can take any value between the vectors $\mathbf{0}$ and $\mathbf{1}$. If an optimal solution to the relaxation is feasible to the mixed integer programming problem, then it is also an optimal solution to the latter [10].

In general, the solution of the linear programming relaxation is a non-integer vector, which means that this solution does not belong to the feasible set of the mixed integer programming problem. Therefore, we construct a suboptimal solution to the relaxation that is feasible to the mixed integer program.

B. Suboptimal solution

Define $(\mathbf{u}^*_1)^f[t] \in \bar{B}^{n^f_{u,1}}$, where $\bar{B} = [0, 1]$, as the optimal control vector to the relaxation for each $t \in T$, which will generally be a non-integer vector between $\mathbf{0}$ and $\mathbf{1}$. A non-integer control vector results in the splitting of friendly resources to several one-step reachable neighboring sectors. An example of a possible optimal solution of the linear programming relaxation is shown in Fig. 1(a).

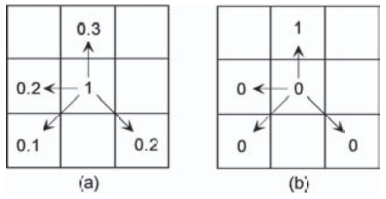

Fig. 1. (a) A possible optimal solution of the linear programming relaxation. (b) The corresponding integer suboptimal solution of the mixed integer programming problem.

Since a vehicle is represented by a binary variable, such a splitting of resources is not feasible. For this reason, among the control quantities that exit from the initial sector or remain to it, we pick up the maximum of them. In Fig. 1(a) this quantity corresponds to the control "0.3". We assign value "1" to this control level, while the rest of them are assigned the value "0". The resulting controls of the suboptimal solution are shown in Fig. 1(b).

In this way, we define an integer control vector that belongs to the feasible set of the mixed integer program, while, in parallel, the sum of the resource levels remains the same as that of the previous time-stage. We call this solution $\tilde{\mathbf{u}}^f_1[t]$, $t \in T$.

C. Receding horizon implementation

Due to the differences between the model used for prediction and optimization, (6) and (7), and the "plant" to be controlled (5), we should expect significant discrepancies between the responses of these two models. To compensate for this approximation, the optimization result will be implemented according to a receding horizon manner [5].

In other words, the following algorithm is implemented:

1) Measure the current state vectors of both teams, $\mathbf{x}^f$ and $\mathbf{x}^e$.
2) Solve the linear programming relaxation of the mixed integer program of (16). Let $(\mathbf{u}_1^*)^f[0]$ be the *first* optimal control of the relaxation.
3) Construct a suboptimal solution, $\tilde{\mathbf{u}}_1^f[0]$, of the initial mixed integer program.
4) Apply only $\tilde{\mathbf{u}}_1^f[0]$ and repeat.

Since the enemy resources, which are the disturbances of the system, are state dependent and the state is bounded, the receding horizon strategy is always stabilizing.

V. Implementation

In this paper, we consider a simplified version of the RoboFlag competition [11], similar to the "RoboFlag drill" [7], where two teams of robots are engaged in an arena of sectors with a region at its center called the defense zone. The defenders' goal is the interception of the attackers, in which case the attackers become inactive, while the attackers' goal is the infiltration of the defenders' protected zone, Fig. 2. Unlike prior work, both teams have equal path planning capabilities using the optimization algorithm proposed here.

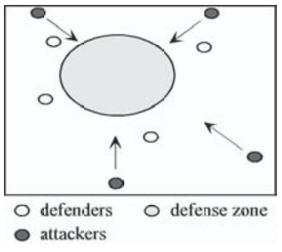

Fig. 2. The RoboFlag drill.

A. Defense path planning

A defense path planning can be designed according to the proposed LP-based path planning. The superscript "d" denotes defenders and will replace superscript "f", while "a" denotes attackers and will replace superscript "e".

Since defenders' objective is the interception of the attackers, the weight α^d must be negative. In addition, a possible reference $\mathbf{X}^d_{\text{ref}}$ for the defenders could be the sectors close to the defense zone, so that the defenders stay always close to it. In the following simulations, we consider the entries of $\mathbf{X}^d_{\text{ref}}$ to be "1" for any sector that belongs to a small zone about the defense zone and "0" otherwise. Moreover, if defenders are not allowed to enter their defense zone, we can set the entries of $\mathbf{X}^d_{\text{obs}}$ to be "1" for sectors in the defense zone and "0" otherwise.

Defense path planning is complete when the stochastic feedback matrix of the attackers, $\mathbf{G}^a$, is determined. The attackers' first priority is the infiltration of the defense zone. Thus, a possible feedback matrix can be one that assigns higher probability to an attacker moving closer to the defense zone. In this case, we can define a function $g^a : N^a(S,R^a) \times R^a \times S \to [0,1]$, such that the probability that the attacker $r_j \in R^a$ will move from sector $s_i \in S$ to sector $s_k \in N^a(s_i,r_j)$ is:

$$g^a_{s_k \leftarrow s_i, r_j} = \frac{\left(\gamma^a_{s_i} + \sum_{s_n}\left\{\gamma^a_{s_n}\right\}\right) - \gamma^a_{s_k}}{(n-1)\cdot\left(\gamma^a_{s_i} + \sum_{s_n}\left\{\gamma^a_{s_n}\right\}\right)} \tag{19}$$

where $\gamma^a_{s_i}$ is the minimum distance from sector s_i to the defense zone, n is the number of one-step reachable destinations (including the current location s_i) and $s_n \in N^a(s_i,r_j)$.

The function g^a satisfies the properties of (9), which implies that a feedback matrix $\mathbf{G}^a$ can be defined according to (8) and (19). Similar functions can be created to include different opponent's objectives.

B. Attack path planning

A similar path planning can be designed for the attackers according to the proposed LP-based path planning. Now the superscript "a" will replace "f", while "d" will replace "e".

The attackers' objective is to enter defenders' protected zone. Therefore, the reference state vector, $\mathbf{X}^a_{\text{ref}}$, is defined such that its entries are equal to "1" if the corresponding sectors belong to the defense zone, and "0" otherwise. At the same time, attackers must avoid defenders, which is encouraged by setting $\alpha^a > 0$.

A stochastic feedback matrix of the defenders, $\mathbf{G}^d$, is also necessary. The defenders' first priority is the interception of the attackers. Assuming that defenders create a probability distribution of the next attackers' locations given by (19), a set of the attackers' most probable future locations can be created, say $L^a[t]$, for each future time $t \in T$.

In this case, we can define a function $g^d : N^d(S,R^d) \times R^d \times S \times T \to [0,1]$, such that the probability that the defender $r_j \in R^d$ will move from sector $s_i \in S$ to sector $s_k \in N^d(s_i,r_j)$ is:

$$g^d_{s_k \leftarrow s_i, r_j}[t] = \frac{\left(\gamma^d_{s_i}[t] + \sum_{s_n}\left\{\gamma^d_{s_n}[t]\right\}\right) - \gamma^d_{s_k}[t]}{(n-1)\cdot\left(\gamma^d_{s_i}[t] + \sum_{s_n}\left\{\gamma^d_{s_n}[t]\right\}\right)} \tag{20}$$

where $\gamma^d_{s_i}[t]$ is the minimum distance from sector s_i to the set of sectors $L^a[t]$, n is the number of one-step reachable destinations (including the current location s_i) and $s_n \in N^d(s_i,r_j)$.

The function g^d satisfies the properties of (9), which implies that a feedback matrix $\mathbf{G}^d$ can be defined according to (8) and (20).

C. Simulations

The efficiency of the LP-based path planning is tested in a RoboFlag drill created in Matlab which involves three attackers and three defenders in an arena of 300 sectors. According to this scenario, an attacker becomes inactive when it is intercepted by a defender.

At the beginning, the LP-based path planning for defense is implemented with $T_p = 6$, $\alpha^d = -1$ and $\beta^d = 0$, which implies that the defenders' priority is only the interception of the attackers. The attackers follow pre-specified paths towards the defense zone that are unknown to the defenders. For this reason, the defenders use a stochastic feedback matrix for the attackers' future locations according to (19).

Fig. 3 shows that the defenders are able to predict the attackers' future locations, while at the same time coordination is achieved, since each defender is assigned to a different attacker. The algorithm runs in 3 *sec per iteration* using Matlab/CPLEX.

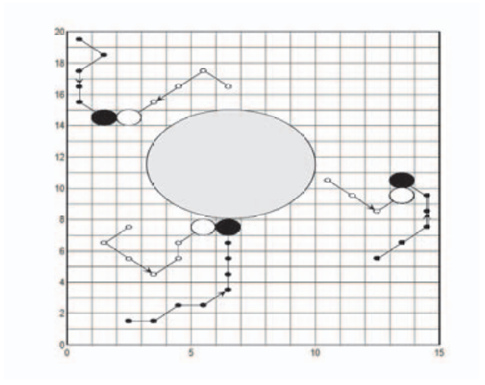

Fig. 3. Defenders optimize their paths according to the LP-based path planning against unknown but pre-specified attackers' paths.

The efficiency of the LP-based path planning was also tested in a more realistic situation, when both defenders and attackers optimize their paths with $T_p = 6$, Fig. 4. The defenders attach weight $\alpha^d = -0.95$ to getting closer to the attackers and weight $\beta^d = -0.05$ to staying closer to the defense zone. On the other hand, the attackers use $\alpha^a = 0.99$ and $\beta^a = -0.01$, which means that they attach more weight to avoiding the defenders than getting closer to the defense zone.

According to Fig. 4, attackers are now able to infiltrate the defense zone. On the other hand, defenders make the attackers follow longer paths towards the defense zone, which means that it is more difficult for the attackers to find a clear path towards the defense zone. Hence, the proposed LP-based algorithm can be used for effective defense and attack planning.

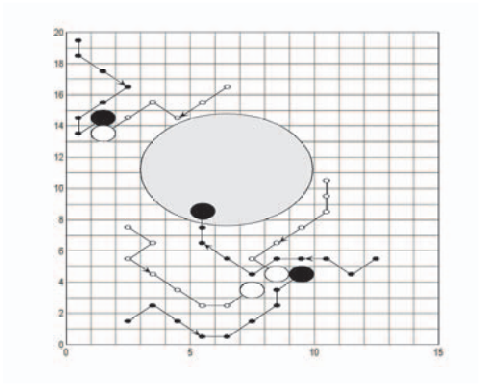

Fig. 4. Both defenders and attackers optimize their paths according to the LP-based path planning.

VI. Conclusions

In this paper, the problem of multi-vehicle coordination in an adversarial environment was formulated as a linear programming optimization. Both friendly and enemy vehicles were modelled as different resource types in an arena of sectors, and the path planning problem was viewed as a resource allocation problem. A simplified model allowed the use of linear programming, while enemy forces were assumed to follow stochastic feedback laws. The solution was implemented according to a receding horizon strategy due to model uncertainties. The utility of the LP-based algorithm was tested in the RoboFlag drill, where both teams of vehicles have equal path planning capabilities using the proposed algorithm. Results showed that the LP-based algorithm can be used for effective multi-vehicle path planning in an adversarial environment.

References

[1] T. W. McLain and R. W. Beard, "Cooperative path planning for timing-critical missions," in *Proc. American Control Conference*, Denver, CO, June 2003, pp. 296–301.

[2] P. R. Chandler and M. Pachter, "Research issues in autonomous control of tactical UAVs," in *Proc. American Control Conference*, Philadelphia, PA, June 1998, pp. 394–398.

[3] A. Dogan, "Probabilistic path planning for UAVs," in *2nd AIAA Unmanned Unlimited Systems, Technologies, and Operations*, San Diego, CA, Sept. 2003.

[4] E. Frazzoli and F. Bullo, "Decentralized algorithms for vehicle routing in a stochastic time-varying environment," in *Proc. IEEE Conf. on Decision and Control*, Paradise Island, Bahamas, Dec. 2004.

[5] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics and constraints," *Automatica*, vol. 35, pp. 407–428, Mar. 1999.

[6] A. Richards, J. Bellingham, M. Tillerson, and J. How, "Coordination and control of multiple UAVs," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, Monterey, CA, Aug. 2002.

[7] M. G. Earl and R. D'Andrea, "Modeling and control of a multi-agent system using mixed-integer linear programming," in *Proc. 41st IEEE Conference on Decision and Control*, Las Vegas, NE, Dec. 2002, pp. 107–111.

[8] M. Flint, M. Polycarpou, and E. Fernandez, "Cooperative path-planning for autonomous vehicles using dynamic programming," in *Proc. IFAC 15th Triennial World Congress*, Barcelona, Spain, 2002.

[9] S. Daniel-Berhe, M. Ait-Rami, J. S. Shamma, and J. Speyer, "Optimization based battle management," in *Proc. American Control Conference*, Arlington, VA, June 2001, pp. 4711–4715.

[10] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization*. Belmont, MA: Athena Scientific, 1997.

[11] R. D'Andrea and R. M. Murray, "The RoboFlag competition," in *Proc. American Control Conference*, Denver, CO, June 2003, pp. 650–655.

[12] G. C. Chasparis, "Linear-programming-based multi-vehicle path planning with adversaries," M.S. thesis, Mechanical Eng. Dept., University of California, Los Angeles, CA, June 2004.