Proceedings of the 1993 International
Symposium on Intelligent Control
Chicago, Illinois, USA - August 1993

Game Theory as a Unifying Structure for a Variety of Robot Tasks

Steven M. LaValle Seth A. Hutchinson
Dept. of Electrical and Computer Engineering
University of Illinois, Urbana, IL 61801

Abstract: *In this paper we propose the use of game theory as a general formalism for representing, comparing, and providing insight into solutions to a wide class of robotics problems. Particularly, we show how game theory can be applied to problems of multiple robot coordination, high-level strategy planning, information gathering through manipulation and/or sensor planning, and pursuit-evasion scenarios. Although some concepts from game theory have been considered in previous robotics contexts, we consider a very general game structure, which consequently has much broader application. We present some preliminary experiments on a two-robot corridor navigation problem in which the robots have independent tasks, and the equilibria in a dynamic game with a rolling time horizon are used for coordination.*

1 Introduction

To perform effectively in real-world settings, robots must be able to plan and execute tasks in uncertain and dynamic environments. Typical sources of uncertainty include sensing accuracy, errors in robot control, changing environments, and discrepancies between geometric object models and physical objects. The problems associated with uncertainty in robotic systems have long been the subject of research in the robotics community. Sharma proposes a probabilistic framework for making local motion plan modifications based on dynamic or partially known workspaces [15]. Takeda and Latombe define a *sensor uncertainty field* which represents the error in sensed robot configuration and is integrated into a mobile robot navigation algorithm [16]. The representation of uncertainty cones for commanded velocity errors, and probabilistic backprojection are discussed in [3, 11, 12]. Other problems and methods are discussed in Section 5. For these types of problems, one needs to sequentially make decisions that achieve a particular task in the presence of problems such as uncertainty, changing environments, and other robots and obstacles.

In this paper, we show how game theory can be applied to problems of multiple robot coordination, high-level strategy planning, information gathering through manipulation and/or sensor planning, and pursuit-evasion scenarios. Game theory is a vast subject which has been explored for decades, and has been applied to problems in economics, politics, gambling, and military strategy. From a modern viewpoint, game theory represents a generalization of decision theory [2], optimal control theory [1], and games considered in AI contexts [18]. Particularly, an extensive amount of material can be found in optimization and control literature; numerous references can be found in [1].

Our motivation for selecting game theory as an appropriate formalism is centered on the following points:

- The game-theoretic concepts provide insight into the nature of problems and suggest solution methods for different robot tasks.

At first glance, it may appear that we are out to encompass all problems in very general game-theoretic terms, leading to an extremely difficult (if not impossible from a computational point of view) problem. Our intentions, however, are to express different robotic tasks as specializations of a general game structure. By accomplishing this, we can consider the overlap with well-studied games that can be efficiently solved, such as the linear-quadratic dynamic game theory presented in [1]. By providing this organizing framework, we can determine which optimization methods are appropriate for a robotic task based on the game formulation. Also, for existing methods that fit into the game-theoretic structure, a number of directions are provided along which the methods might be generalized.

- Game theory provides a rigorous formulation of the tradeoff between planning and control.

There are static games in which the players each plan their strategies in advance by considering the space of possible outcomes, and a strategy is implemented as pure planning that does not take into account information feedback that could occur during the game execution. On the other hand, as game theory represents a generalization of optimal control, an information feedback structure can be arranged to make the game into a multiple-agent control scenario.

- The game concepts and terminology precisely define many relevant aspects of situations in which some form of competition or conflict occurs.

As an example, the concept of cooperation in a game has been carefully studied. Robots behaving in a cooperative manner may accomplish their independent goals by agreeing on a strategy of mutual sacrifice that produces the lowest possible combined cost. On the other hand, in a noncooperative situation, each robot must consider the fact that the others are attempting to independently solve their own goals. Each robot will have to take into account possibilities that other robot actions can directly conflict with its goals, leading to an equilibrium solution. Section 2 provides more details on cooperation, equilibria, and other game concepts.

- As a generalization of decision theory, game theory encourages the use of statistical modeling at a variety of levels.

As pointed out in [7, 10] statistical modeling and estimation are important concerns for experience-based and sensor-based robotic applications, and game theory provides a natural framework for describing sequences of statistical decisions.

- Even when such solutions cannot be practically obtained, the formalism can provide a point of reference to which approximate or partial solutions can be compared.

This concept is similar to what often occurs with Bayesian decision theory. The theory itself provides a formal specification of optimal decision making, and other methods, such as nearest-neighbor classifiers, can be assessed in relation to the optimal Bayes' decision rule [4]. The use of a rolling horizon in Section 4 is an example of an approximate game solution.

In Section 2 we define and describe some important concepts from game theory. Section 3 introduces a game formulation that applies in a general robotics context. In Section 4 we apply the formulation of Section 3 to the specific problem of two competing robots that are navigating in a corridor system, each attempting to accomplish specific tasks. Some preliminary experimental examples are presented in Section

0-7803-1206-6/93/$3.00 ©1993 IEEE

4.1 which illustrate the utility of the game theory concepts. Section 5 describes how the general formulation applies to specific problems that have been previously considered in robotics research. Finally, Section 6 summarizes our early conclusions and discusses the areas that we intend to explore in the future.

2 Game Theory Overview

There is a vast wealth of literature on game theory, and rather than provide a complete survey we provide a discussion of some of the key principles. A more detailed presentation can be found, for instance, in [1].

In a game, the *players* are the participants in the game. Each player has an *action set* available when that player is allowed to make a move during the play of a game. The number of *stages* in a game refers to the number of times that one or more players are called upon to make decisions.

The *state* of the game encodes information about the configuration of the game at some stage. The application of player actions at a stage produces a new state, which is specified through a *state-transition equation.*

A *player strategy* is a specification of the action that the player will take for each possible situation that the player could be confronted with during a game. This could be a deterministic specification of the action that a player will choose, termed a *pure strategy*, or could be the specification of a probability distribution over each of the action sets, termed a *mixed strategy.* With a pure strategy, the player will make the same decisions every time the game is played; however, when a mixed strategy is used, the player selects a move by randomly sampling from the probability distribution. Hence the particular actions of the player can vary over different game executions. The set of strategies for all players form a *game strategy.*

The motivation for the selection of certain actions by a player is effected by defining a real-valued *loss function* on the strategy space for the game. This corresponds to the amount of penalty (or inversely, reward) that is given to the player if the game is played and terminates in a certain manner. It can depend on actions taken by any of the players and also on the game state, at any stage.

The effects of noise or other uncontrollable actions can be modeled by addition of a new player called *nature* [2]. Nature is assumed to take a mixed strategy, thereby probabilistically influencing the game state at various stages of play. The others players are forced to take the effects of nature into account when selecting strategies.

The notion of a solution of a game requires some further concepts. Since there are in general many independent players in a game, the idea of a better game strategy is more complex than that for an ordinary optimization problem. If we consider a one-player game (or possibly a game with one true player and nature) the best player strategy would be the set of actions that produce the minimal loss (or expect to produce minimal loss). However, when other players are involved, some game strategies may be very favorable for one player, but poor for another.

The concept of player cooperation therefore becomes important when considering favorable game outcomes. If the players agree to choose their actions in unison, so that a mutually beneficial outcome can be obtained, we have a *cooperative game.* Games of this type can usually be cast as an optimization problem in which some function of the player loss functions is optimized. Otherwise the players act in ways that take into account the conflicting interests with the other players, referred to as a *noncooperative game*, and attempt to minimize the loss accordingly. The most extreme case of conflicting interest is a *zero-sum* game, in which two players interests are diametrically opposed. The "solution" to a game of the noncooperative type is referred to as an *equilibrium* since it represents a game strategy that provides a balance between the independent interests of the players. For the experiments presented in this paper we consider Nash equilibria [1].

The discussion thus far has been limited to what are called *static games.* Although there are multiple decisions being made, which may appear dynamic, the set of strategies of each player can be viewed as one large space of decisions. The selection of a strategy corresponds to selecting an element from this space of decisions.

A more general formulation of games, termed *dynamic games*, takes into account several new issues. The most important aspect of dynamic games is the amount of knowledge that each player has about the current and previous game states, and the previous actions of other players. In general this is considered as a form of information feedback, and hence partially explains why game theory generalizes optimal control theory. A *behavioral strategy* is the specification of the player action, conditioned only on the space of information that could be available to the player at a game stage. In a strategy of this type, the player will make the same decisions in game states it cannot distinguish between due to its incomplete information.

3 A General Game-Theoretic Formulation of Robotic Tasks

We now present the components involved in the general game-theoretic formulation and indicate their intended use in a robotics context. Sections 4 and 5 will discuss the specialization of this formalism to particular areas. The components presented here constitute a modified form of the general *discrete-time infinite dynamic game* presented in [1].

We have the following components:

1. An index set, $\mathbf{N} = \{1, 2, \ldots, N\}$, of N players.

   We consider three different types of players: robots denoted by $\mathcal{A}_i$, obstacles denoted by $\mathcal{B}_i$, and nature. The robots are the players of the game in the sense discussed in Section 2. Each $\mathcal{A}_i$ is attempting to choose some actions that will appropriately accomplish its goals, while coping with the conflicts that arise due to the other players. The set of robot indices will be referred to as $\mathbf{N}_{\mathcal{A}}$. The obstacles are special players that have their strategies fixed (i.e, they are not attempting to select actions to achieve some goal, but have a fixed pure or mixed strategy in use). Finally, nature represents noise and uncertainty that can arise during the game execution. Of course, the consideration of obstacles and nature is optional.

2. An index set, $\mathbf{K} = \{1, 2, \ldots, K\}$ denoting the *stages* of the game, in which K (possibly infinite) is the maximum possible number of moves a robot or obstacle is allowed to make in the game.

   Although we are fixing the number of stages in this formulation, we can also allow early termination of the game, for instance if all of the robots have detected that they have accomplished their goals. In a limiting case, one can consider a continuum of stages, which ultimately yields a system of differential equations that specify the game.

3. A set, X, with some topological structure called the *state space.* The state of the game, x_k at a stage in $\mathbf{K} \cup \{K+1\}$, belongs to X.

   This state space could be considered as a *configuration space*, which has been used extensively in robot motion planning literature to represent the position(s) of robot(s) [11], or as a *phase space* in nonholonomic planning planning; however, this state space could also represent some other information relevant to the game situation.

4. A set, U_k^i, with some topological structure, defined for each $k \in \mathbf{K}$ and $i \in N_{\mathcal{A}}$, which is called the *action set* of $\mathcal{A}_i$ at stage k.

   For each robot at each possible stage, this set characterizes the actions that can be taken, which have some consequence on the game state. These actions could be

prespecified, or could be dependent on the game state at stage k.

5. A function $f_k : X \times U_k^1 \times \ldots \times U_k^N \rightarrow X$ defined for each $k \in \mathbf{K}$ so that

$$x_{k+1} = f_k(x_k, u_k^1, \ldots, u_k^N), k \in \mathbf{K}, \quad (1)$$

and some $x_1 \in X$ is designated the initial state of the game. One might also consider states that cause the termination of the game. The difference equation above is called the *state transition equation* of the game.

This equation defines the effect of the player actions on the game state. The initial state, x_1, could be given initially, and the player actions deterministically produce subsequent states. One could also consider a stochastic situation in which a probability distribution on the state space is given initially, and the actions incrementally transform the probability distribution. This is a generalization of the situation encountered in Markov process control [8, 20]. In a limiting case with a continuum of stages, the state transition equation becomes a differential equation; this is included in the study of differential games [1].

6. A set Y_k^i with some topological structure, defined for each $k \in \mathbf{K}$ and $i \in \mathbf{N}_\mathcal{A}$, and called the *sensor space* of $\mathcal{A}_i$ at stage k, to which the sensed value y_k^i of $\mathcal{A}_i$ belongs at stage k.

The set Y_k^i represents the set of all values that some measurement system could obtain by sensing the state of the world. For instance, with a proximity sensor Y_k^i could represent a set of positive real values. If the sensor is an intensity image, then Y_k^i represents the space all arrays of intensity values (with more practical consideration, one may extract features from the image, and Y_k^i would represent the feature space).

7. A function $h_k^i : X \rightarrow Y_k^i$, defined for each $k \in \mathbf{K}$ and $i \in \mathbf{N}_\mathcal{A}$, so that

$$y_k^i = h_k^i(x_k), k \in \mathbf{K}, i \in \mathbf{N}_\mathcal{A} \quad (2)$$

which is the *state-measurement equation* of $\mathcal{A}_i$ concerning the value of x_k.

This is the common projection from the world state to a sensor space that has been investigated in the robotics and computer vision literature [5, 9, 10]. This also represents the standard information feedback structure that occurs in control theory.

8. A set η_k^i, defined for each $k \in \mathbf{K}$ and $i \in \mathbf{N}_\mathcal{A}$ as a subset of all actions made by any player at any previous stage and any state-measurements made by robots at any stage.

This determines the information gained and recalled by $\mathcal{A}_i$ at stage k of the game. Specification of η_k^i for each $k \in \mathbf{K}$ characterizes the *information structure* of $\mathcal{A}_i$, and the collection over $i \in \mathbf{N}_\mathcal{A}$ of these information structures is the *information structure* of the game.

9. The set of all possible values for η_k^i is denoted by N_k^i and is called the *information space* for $\mathcal{A}_i$ at stage k.

The set N_k^i represents the set of all distinguishable sets of information that could be received by $\mathcal{A}_i$ during stage k of the game. To specify a strategy for $\mathcal{A}_i$, one only needs to specify which actions to take for the various elements in the information space.

10. A prespecified class Γ_k^i of mappings $\gamma_k^i : N_k^i \rightarrow U_k^i$ which are the permissible *strategies* of $\mathcal{A}_i$ at stage k. The aggregate mapping $\gamma = \{\gamma_1^i, \gamma_2^i, \ldots, \gamma_K^i\}$ is a *strategy* for $\mathcal{A}_i$ in the game, and the class Γ^i of all such mappings γ^i so that $\gamma_k^i \in \Gamma_k^i, k \in \mathbf{K}$, is the *strategy space* of $\mathcal{A}_i$.

Thus when $\mathcal{A}_i$ receives the particular information $\eta_k^i \in N_k^i$, the action $\gamma_k^i(\eta_k^i)$ will be taken. One could also consider γ_k^i as a probability distribution on U_k^i that is conditioned on η_k^i, to yield a mixed strategy.

11. A functional $L^i : (X \times U_1^1 \times \ldots \times U_1^N) \times (X \times U_2^1 \times \ldots \times U_2^N) \times \ldots \times (X \times U_k^1 \times \ldots \times U_k^N) \rightarrow \Re$ defined for each $i \in \mathbf{N}_\mathcal{A}$, and called the *loss functional* of $\mathcal{A}_i$ in the game.

The loss functional essentially specifies the goals of the robot players. Examples of loss functionals will be discussed in Sections 4.1 and 5.

At this point, no consideration has been given to obstacles, $\mathcal{B}_i$. These can be considered in two different ways (assuming obstacle avoidance is desired). The state space can be constrained to prohibit the contact of robots and obstacles in a workspace, or alternatively, the loss functionals can be defined to assign extremely high loss to states in which robots and obstacles are in contact.

4 Coordination of Multiple Robot Tasks

In this section we consider the coordination of two robots in a complex corridor system. The robots have independent goal locations and initial locations. The conflicts arise when robots need to occupy the same corridors at the same time instances along their optimal paths. Through the use of game theory, we can obtain solutions that are mutually beneficial, although suboptimal for each individual robot. A discussion of related problems and approaches can be found [11]

First we define a directed graph, $G = (V, E)$, in which V represents the set of vertices and E the set of edges. The vertices correspond to locations that a robot can occupy. The edges correspond to locations that a robot can move to through the application of an action.

The following game formulation applies to two robots; however, an n robot situation is straightforward to consider.

1. There are two players, $\mathcal{A}_1$ and $\mathcal{A}_2$, which are robots.
2. The number of stages, K, is finite; however, early game termination can occur at some stage $k \leq K$.
3. The state space, X, is represented by the set of all pairs of vertices (i.e., of the form (v^1, v^2)), in which each element of a pair represents the location of one of the robots.
4. The action set, u_k^i, available to $\mathcal{A}_i$ at a stage k, represents the set of vertices adjacent to v_k^i. Hence the action sets depend on x_k.
5. The configuration-transition equation is

$$x_{k+1} = (n(v_k^1, u_k^1), n(v_k^2, u_k^2)) \quad (3)$$

in which n represents the vertex that is reached (through the appropriate edge in E) when applying u_k^i to the vertex v_k^i.

6. There is perfect information in the game (i.e., the robots are aware of the game state, previous states, and previous actions at every stage of the game).
7. There is a *stage-additive* loss functional. By this we mean that for $\mathcal{A}_i$ at stage k the loss can be expressed as

$$L^i(\gamma^1, \gamma^2) = \sum_{j=1}^{k} L_j^i(x_j, u_j^1, u_j^2). \quad (4)$$

In the expression above we use γ^i to refer to the sequence of actions taken by $\mathcal{A}_i$ up to and including stage k.

We also have a termination condition that stops the game if both robots are at some prespecified goal vertices.

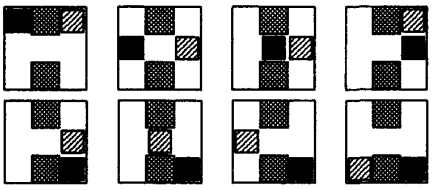

Figure 1. A cooperative solution, yielding a resulting loss of $(31, 53)$. In this and subsequent figures, robot $\mathcal{A}_1$ is black, and $\mathcal{A}_2$ is striped.

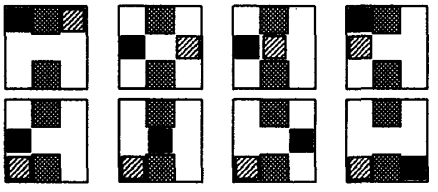

Figure 2. The symmetrically equivalent cooperative solution, yielding a resulting loss of $(53, 31)$.

4.1 Coordination Experiments

The particular loss functional used in our experiments is defined as

$$L_j^i(x_j, u_j^1, u_j^2) = c(u_j^i) + d^i(v_j^i) \tag{5}$$

in which $c(u_j^i)$ represents the cost associated with taking action u_j^i. We assume this cost is stage independent, and identical for both robots. For simplicity we assign a cost of 1 for an action that remains at a vertex (a cost for wasting a stage) and a cost of 2 to move to an adjacent vertex. The term $d^i(v_j^i)$ is the cost associated with having robot $\mathcal{A}_i$ at vertex v_j^i (which is a part of x_j^i). We take this to be twice the distance to the goal (in terms of number of actions required along the optimal path), which is precomputed for each vertex and robot using Dijkstra's algorithm. We associate infinite cost to situations in which both robots occupy the same vertex.

For a given strategy, γ, let (l_1, l_2) represent the cumulative losses at the end of the game for $\mathcal{A}_1$ and $\mathcal{A}_2$. We can consider a partial ordering on strategies by defining $\gamma \preceq \gamma'$ if $l_1 < l_1'$ and $l_2 < l_2'$. We can then refer to a set of *maximal strategies* as those that are maximal with respect to $\preceq$.

First we consider a 7-vertex game, describing corridors arranged in an "H" shape, shown in Figure 1. The robots consider moving between 7 quantized positions, which are square-shaped cells. The game state on the upper left in Figure 1 shows the initial configuration; the game state on the lower right is the goal configuration. Conflict arises since the robots need to occupy the center cell at the same instant in time to optimally accomplish their goals.

The sequences of game states that appear in Figures 1 and 2 represent two maximal strategies obtained by setting $K = 7$. The robot that waits outside the middle cell oscillates for a few steps since the vertices closer to the goal have lower loss. The problem leads to nonunique equilibria due to symmetry. In a noncooperative situation, if $\mathcal{A}_1$ selects the second strategy and $\mathcal{A}_2$ selects the first strategy, the two robots will wait for a while and then collide in the middle. Likewise, if $\mathcal{A}_1$ chooses the first one and $\mathcal{A}_2$ chooses the second one, the robots will both move directly toward the tunnel, leading to another collision. Therefore, in this case some cooperation is required to guarantee that the robots will accomplish their goals.

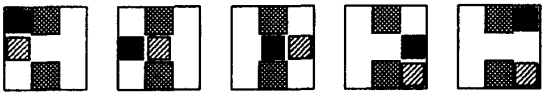

Figure 3. A unique maximal Nash equilibrium solution with loss $(35, 40)$.

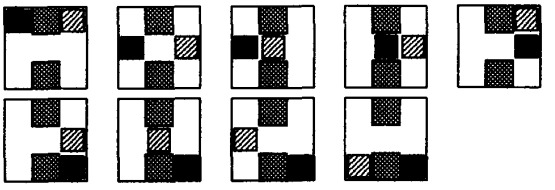

Figure 4. A solution obtained from using the rolling horizon procedure with $K' = 1$. The resulting loss is $(38, 60)$.

In Figure 3 we show a situation that leads to a unique equilibrium; however, the same level of conflict does not arise since both robots can optimally accomplish their tasks.

The next strategies were obtained using a rolling horizon technique [8], since the size of the strategy space grows dramatically as the number of stages increases. Also, we would like to avoid the need to specify some arbitrary stage limit, so that one would not be required to be certain that the goal can be reached with the number of given stages. This technique illustrates how an approximate technique can be obtained from the general game formulation.

We select some number of stages, K', which is assumed to be much smaller than the number of stages required to solve the game. A strategy for the original game is obtained through the following procedure:

1. Begin with stage $k = 1$.
2. Select a strategy, γ from the game obtained by considering the actions from stage k to $k + K'$.
3. Execute the first action of γ for each robot.
4. If the goal state has not been reached, then $k := k + 1$, and go to Step 2.

In Step 2 we select the maximal strategy that produces the least total cost (summing the loss functions of the players). Hence, in this procedure we use a cooperative approach which leads to a unique solution. The resulting strategy to the original game is specified by the sequences of actions that are executed in Step 3.

If we use the rolling-horizon procedure on the problem in Figures 1 and 2, we obtain one of the maximal Nash equilibria if $K' \geq 2$. If we take $K' = 1$, then a suboptimal strategy (but reasonable) is obtained, as shown in Figure 4.

The final experiment was performed on a significantly harder problem, having 122 vertices. Figure 5 shows results from taking K' as 3. The results are similar when $K' = 2$, but when a 3-stage horizon was used, the problem was solved in one less step, resulting in a better strategy in terms of total loss. If we take $K' = 1$, however, the game ends up in an oscillation because the robots cannot consider moves far enough ahead that will resolve the conflict. Therefore, the last two stages repeat indefinitely.

5 Other Problem Classes to which Game Theory Applies

This section describes three particular applications of the formulation presented in Section 3. Sections 5.1 and 5.3 describe how previously proposed problems fit into the game theoretic framework, and Section 5.2 describes a game specification that we are proposing.

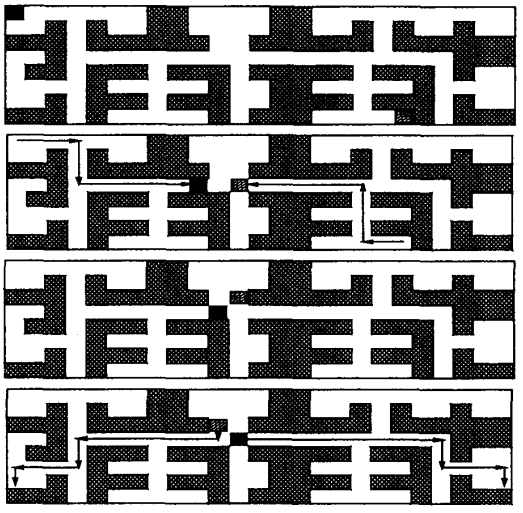

Figure 5. A solution obtained from using the rolling horizon procedure with $K' = 3$. The resulting loss is (990, 818).

5.1 Stochastic Robot Manipulation

In this section we discuss how stochastic robot manipulation fits into the game theoretic framework. We point out, however, that this problem has already been viewed in the literature as a one-robot game against nature [17]; therefore, we will briefly describe how the work relates to our framework. We will in particular describe the framework proposed by Goldberg [7].

We consider a two-player game with one robot $\mathcal{A}_1$ and nature. Nature acts in stage 1, and $\mathcal{A}_1$ acts alone in each subsequent stage. Consider an unlimited number of stages (in practice, of course, early termination will occur after a finite number of stages). Consider a finite state space, X, which corresponds to a discretized representation of the position and/or orientation of an object to be grasped. There is a stage-independent finite action set, U^1, representing a set of operations that can be performed by the robot. In [7] this corresponds to quantized orientations of a parallel gripper which is attempting to grasp the object.

The state-transition equation is specified stochastically. For a given action, u_k^1 and state x_k, a probability distribution, $P(x_{k+1}|u_k^1(x_k))$ is specified on the state space reflecting the effect of applying the operation (this distribution is termed a hyperstate in [7]).

The only information available to $\mathcal{A}_1$ during the play of the game is the sequence of actions that it has applied thus far. In this sense, the planning strategy is sensorless. In stage 1, nature selects some $x_1 \in X$, which corresponds to the placement of the object in some configuration. Although $\mathcal{A}_1$ does not observe this placement, the mixed strategy of nature is known, and hence a probability distribution over X is known. The operator effects are specified with a state-transition matrix for each action u^1, which computes the resulting probability distribution on X, given an action and initial probability distribution on X. A loss function (termed a real-valued cost metric) is specified, which is a function of an action and the resulting distribution, and measures the amount of uncertainty in X. A backchaining algorithm is presented which determines a sequence of actions that produces a sufficiently small amount of uncertainty.

5.2 High-level Strategy Planning

In this section we consider a problem in which a robot has at its disposal some finite set of motion planning strategies, and a motion planning problem (planning a path from a starting point to an ending point in configuration space) is drawn from a space of problems $\mathcal{M}$. The robot is confronted with the task of selecting a method (or methods) that will lead to the solution of the problem, and is given statistical information about the probability of success of each method and its expected cost. This type of high-level strategy planning is similar to the case-based reasoning approach addressed in [13], and the problem is similar in form to that of [6].

Using our framework we will use nature to model two different aspects. When a problem is proposed for the robot to solve, we consider it as having been sampled from a distribution of problems. Thus at the first stage of the game, the problem selection will be considered as an action by nature. Nature will also be used to model the probability of a particular method succeeding. At various stages of the game, the robot has the probabilities that certain methods will succeed, and the success/failure outcome is considered as an action by nature.

Hence, we consider a game with two players, $\mathcal{A}_1$ and nature. We select K, the number of stages, very large, and we expect the game to terminate long before K stages. The state space is the cartesian product

$$X = \mathcal{M} \times \{U, S\} \tag{6}$$

in which U and S are literals representing "unsolved" and "solved." We allow nature to act only at odd-numbered stages, and $\mathcal{A}_i$ to act in the remaining stages. The action set U^1 is the set of motion planning strategies that $\mathcal{A}_i$ can attempt. We drop the k subscript to indicate that this action set is not stage dependent (except for the fact that it only pertains to even-numbered stages). For nature we have $U_1^2 = \mathcal{M}$, and $U_k^2 = \{U, S\}$ when $k > 1$ and k is odd.

We will specify the state-transition equation by describing the play of the game. At stage 1 nature selects a motion planning problem $m \in \mathcal{M}$ and the game state is (m, U), meaning that problem m is unsolved. In stage 2, $\mathcal{A}_1$ selects a motion planning strategy, u_2^1. In stage 3, nature randomly samples from $\{U, S\}$. If nature chooses S, then the game state becomes (m, S), which causes the game to terminate. Otherwise the game state remains at (m, U) and we progress to stage 4, which is similar to stage 2. Hence, $\mathcal{A}_1$ continues to try to apply actions that place the game in the "solved" portion of the state space.

We define the sensor space, Y^1, as the target set of a vector-valued function, h^1 of $\mathcal{M}$ (technically we should have a function of X; however, the sensor space will only be defined if m is unsolved). The sensor space has the same interpretation as a feature space in statistical pattern recognition [4]. Rather than making decisions directly from $\mathcal{M}$, a projection is formed onto a feature space for computational expediency, which is given in our case by h^1. The information space, N^1, is generated by the sensor space Y^1 and the set of previous actions made by $\mathcal{A}_1$ and nature. We also take $N^1 = N^2$. The strategy space for $\mathcal{A}_1$ specifies the decisions that are made given different feature values and different sets of previous actions. The strategy space for nature specifies probability distributions over $\{U, S\}$ given the various points in N^2.

The loss functional that guides the actions of $\mathcal{A}_1$ is just the total time cost of the motion planning strategies that were executed:

$$L^1 = \sum_{i=1}^{k} cost(u_k^1). \tag{7}$$

To apply this formulation in a practical situation, we must address several issues. The *sensor space* must be defined for which $h^1(m)$ can be efficiently computed and appropriate, informed decisions can be made by $\mathcal{A}_1$. The extraction of useful features for a motion planning problem is a challenging problem, which has been partially addressed in [13]. One might also want to work the cost of feature computation directly into the loss functional, and select only those features that

lead to minimal cost. Note that we could also play the game without state feedback, and use the probabilities of success or failure in general; however, one would expect the solutions to be poorer. Another issue is that $\mathcal{A}_1$ will make its decisions based on the mixed strategy of nature, which must be determined. This could, for instance, be determined by constructing a cellular partition of Y^1, and using the statistical average of successes vs. total trials in each cell.

5.3 Pursuit and Evasion Scenarios

Both the problems of pursuit and evasion are important in robotics contexts. The problem of pursuit is also referred to as tracking. The evasion problem can be considered as keeping multiple robots and/or obstacles from colliding while executing tasks, and has been considered in [14]. Examples of game-theoretic solutions applied to pursuit and evasion scenarios can be found in [1, 19].

In this section we discuss how our formalism encompasses the particular pursuit framework proposed in [10]. In this work, a *temporal belief network* is used to graphically represent a sequential decision process, which in turn can be embedded within the components of Section 3.

We consider a two-player game with one robot, $\mathcal{A}_1$, and one moving obstacle $\mathcal{B}_1$. We refer to the second player as an obstacle since we will not be determining a strategy for that player, but one could alternatively consider each of them as robots. We consider a K-stage game, since a finite number of fixed points in time are used in [10]. The set of locations of the robot and obstacle are quantized into finite sets of manageable size. The state space in our game is the cartesian product of robot and obstacle locations (referred to as S_R and S_O in [10]). At any point in time, there is a finite set of actions U_k^1 available to the robot (referred to as A_R). Robot actions cause a change in the game state; however, the exact positions of $\mathcal{A}_1$ and $\mathcal{B}_1$ are unknown to $\mathcal{A}_1$, and information is obtained through sensor information. We can use Y_k^1 and Y_k^2 (referred to as O_R and O_T in [10]) to describe the projection that takes places through the application of an abstract sensor mapping. The loss functional takes the form of a value function that reflects the uncertainty in the expected location in the target.

6 Future Pursuits

We consider the work presented in this paper as an early investigation into a formalism that we intend to utilize extensively in future work. From the formulations and discussions presented in Section 5 we have shown that our formalism applies to a wide class of problems. Further, our initial experiments show that game-theoretic concepts provide useful insight into a particular multiple-robot planning problem.

In future work we intend to:

- Consider coordination experiments with more robots and/or stochastic models.
- Develop methods of transforming geometric planning problems into a game state space.
- Further explore the game scenarios from Section 5 and consider other problem classes.
- Consider other game equilibria, such as Stackelberg.

Acknowledgements

This work was sponsored by NSF under grant #IRI-9110270. We thank Garry Didinsky for a helpful discussion. We are also grateful to Becky Anderson, Jon Gratch, and Rajeev Sharma for providing many helpful comments and suggestions.

References

[1] T. Başar and G. J. Olsder. *Dynamic Noncooperative Game Theory.* Academic Press, London, 1982.

[2] D. Blackwell and M. A. Girshnik. *Theory of Games and Statistical Decisions.* Dover Publications, New York, NY, 1979.

[3] R. C. Brost and A. D. Christiansen. Probabilistic analysis of manipulation tasks: A research agenda. In *IEEE International Conference on Robotics and Automation,* pages 3:549–556, 1993.

[4] P. A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach.* Prentice/Hall Publications, Englewood Cliffs, NJ, 1982.

[5] B. R. Donald and J. Jennings. Sensor interpretation and task-directed planning using perceptual equivalence classes. In *IEEE International Conference on Robotics and Automation,* pages 190–197, Sacramento, CA, April 1991.

[6] D. Einav and M. R. Fehling. Computationally-optimal real-resource strategies for independent, uninterruptible methods. In *Uncertainty in Artificial Intelligence 6,* pages 145–158. North-Holland, Amsterdam, 1991.

[7] K. Y. Goldberg. *Stochastic Plans for Robotic Manipulation.* PhD thesis, Carnegie-Mellon, Pittsburgh, PA, August 1990.

[8] O. Hernández-Lerma and J. B. Lasserre. Error bounds for rolling horizon policies in discrete-time markov control processes. In *IEEE Trans. on Automatic Control,* pages 1118–1124, 1990.

[9] K. Ikeuchi and T. Kanade. Modeling sensors: Toward automatic generation of object recognition program. *Comp. Vision, Graphics, and Image Process.,* 48:50–79, 1989.

[10] J. Kirman, K. Basye, and T. Dean. Sensor abstraction for control of navigation. In *IEEE International Conference on Robotics and Automation,* pages 2812–2817, 1991.

[11] J. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, MA, 1991.

[12] T. Lozano-Pérez, M. T. Mason, and R. H. Taylor. A simple motion planning algorithm for generating robot manipulators. *Int. J. of Robot. Res.,* 3(3):224–238, 1984.

[13] S. Pandya and S. A. Hutchinson. A case-based approach to robot motion planning. In *Proc. IEEE Int. Conf. on Syst., Man, Cybern.,* pages 492–497, Chicago, October 1992.

[14] J. H. Reif and M. Sharir. Motion planning in the presence of moving obstacles. In *Proc. of IEEE Symp. on Foundat. of Comp. Sci.,* pages 144–154, 1985.

[15] R. Sharma. A probabilistic framework for dynamic motion planning in known environments. In *IEEE International Conference on Robotics and Automation,* pages 2459–2464, Nice, France, May 1992.

[16] H. Takeda and J. Latombe. Sensory uncertainty field for mobile robot navigation. In *IEEE International Conference on Robotics and Automation,* pages 2465–2472, Nice, France, May 1992.

[17] R. H. Taylor, M. T. Mason, and K. Y. Goldberg. Sensor-based manipulation planning as a game with nature. In *Fourth International Symposium on Robotics Research,* pages 421–429, 1987.

[18] P. H. Winston. *Artificial Intelligence.* Addison-Wesley, Reading, MA, 1992.

[19] J. Yong. On differential evasion games. *SIAM J. Control and Optimization,* 26(1):1–22, January 1988.

[20] Q. Zhu. Hidden Markov model for dynamic obstacle avoidance of mobile robot navigation. *IEEE Trans. on Robotics and Automation,* 7(3):390–397, June 1991.