Chapter 1

Introduction

Steven M. LaValle
University of Illinois

Copyright 2004
Chapter 1

Introduction

1.1 Planning to Plan

Planning is a term that means different things to different groups of people. A fundamental need in robotics is to have algorithms that can automatically tell robots how to move when they are given high-level commands. The terms motion planning and trajectory planning are often used for these kinds of problems. A classical version of motion planning is sometimes referred to as the Piano Mover’s Problem. Imagine giving a precise 3D CAD model of a house and a piano as input to an algorithm. The algorithm must determine how to move the piano from one room to another in the house without hitting anything. Most of us have encountered similar problems when moving a sofa or mattress up a set of stairs. Robot motion planning usually ignores dynamics and other differential constraints, and focuses primarily on the translations and rotations required to move the piano. Recent work, however, does consider other aspects, such as uncertainties, differential constraints, modeling uncertainties, and optimality. Trajectory planning usually refers to the problem of taking the solution from a robot motion planning algorithm and determining how to move along the solution in a way that respects the mechanical limitations of the robot.

Control theory has historically been concerned with designing inputs to systems described by differential equations. These could include mechanical systems such as cars or aircraft, electrical systems such as noise filters, or even systems arising in areas as diverse as chemistry, economics, and sociology. Classically, control theory has developed feedback policies, which enable an adaptive response during execution, and has focused on stability, which ensures that the dynamics do not cause the system to become wildly out of control. A large emphasis is also placed on optimizing criteria to minimize resource consumption, such as energy or time. In recent control theory literature, motion planning sometimes refers to the construction of inputs to a nonlinear dynamical system that drives it from an initial state to a specified goal state. For example, imagine trying to operate a remote-controlled hovercraft that glides over the surface of a frozen pond. Suppose we would like the hovercraft to leave its current resting location and come to rest at another specified location. Can an algorithm be designed that computes the desired inputs, even in an ideal simulator that neglects uncertainties that arise from model inaccuracies? It is possible to add other considerations, such as uncertainties, feedback, and optimality, but the problem is already challenging enough without these.

In artificial intelligence, the terms planning or AI planning take on a more discrete flavor. Instead of moving a piano through a continuous space, as in the robot motion planning problem, the task might be to solve a puzzle, such as the Rubik’s cube or a sliding tile puzzle, or to achieve a task that is modeled discretely, such as building a stack of blocks. Although such problems could be modeled with continuous spaces, it seems natural to define a finite set of actions that can be applied to a discrete set of states, and to construct a solution by giving the appropriate sequence of actions. Historically, planning has been considered different from problem solving; however, the distinction seems to have faded away in recent years. In this book, we do not attempt to make a distinction between the two. Also, substantial effort has been devoted to representation language issues in planning. Although some of this will be covered, it is mainly outside of our focus. Many decision-theoretic ideas have recently been incorporated into the AI planning problem, to model uncertainties, adversarial scenarios, and optimization. These issues are important, and are considered here in detail.

Given the broad range of problems to which the term planning has been applied in the artificial intelligence, control theory, and robotics communities, one might wonder whether it has a specific meaning. Otherwise, just about anything could be considered as an instance of planning. Some common elements for planning problems will be discussed shortly, but first we consider planning as a branch of algorithms. Hence, this book is entitled Planning Algorithms. The primary focus is on algorithmic and computational issues of planning problems that have arisen in several disciplines. On the other hand, this does not mean that planning algorithms refers to an existing community of researchers within the general algorithms community. This book will not be limited to combinatorics and asymptotic complexity analysis, which is the main focus in pure algorithms. The focus here includes numerous modeling considerations and concepts that are not necessarily algorithmic, but aid in modeling, solving and analyzing planning problems.

The obvious goal of virtually any planning algorithm is to produce a plan. Natural questions are: What is a plan? How is a plan represented? What is it supposed to achieve? How will its quality be evaluated? Who or what is going to use it? Regarding the user of the plan, it clearly depends on the application. In most applications, an algorithm will execute the plan; however, sometimes the user may be a human. Imagine, for example, that the planning algorithm provides you with an investment strategy. A generic term that will used frequently here to refer to the user is decision maker. In robotics, the decision maker is simply referred to as a robot. In artificial intelligence and related areas, it has become
1.2. ILLUSTRATIVE PROBLEMS

This is the only section that still needs to be written. Currently, it only has a couple of pasted examples.

Suppose that we have a tiny mobile robot that can move along the floor in a building. The task is to determine a path that it should follow from a starting location to a goal location, while avoiding collisions. A reasonable model can be formulated by assuming that the robot is a moving point in a two-dimensional environment that contains obstacles.

Let \( W = \mathbb{R}^2 \) denote a two-dimensional world which contains a point robot, denoted by \( A \). A subset, \( O \), of the world is called the obstacle region. Let the remaining portion of the world, \( W \setminus O \) be referred to as the free space. The task is to design an algorithm that accepts an obstacle region defined by a set of polygons, an initial position, and a goal position. The algorithm must return a path that will bring the robot from the initial position to the goal position, while only traversing the free space. Algorithms that find exact solutions to this problem are given in Section 6.2.

Figures 1.2 and 1.3 show considerably more challenging problems.

1.3. Basic Ingredients of Planning

Although the subject of this book spans a broad class of models and problems, there are several basic ingredients that arise throughout virtually all of the topics covered as part of planning.

**State:** Planning problems will involve a state space that captures all possible situations that could exist. The state could, for example, represent the configuration of a robot, the locations of tiles in a puzzle, or the position and velocity of a helicopter. Both discrete (finite, or countably infinite) and continuous (uncountably infinite) state spaces will be allowed. One recurring theme through most of planning is that the state space will usually be represented implicitly by a planning algorithm. In most applications, the size of the state space (in terms of number of states or combinatorial complexity) is much too large to be explicitly represented. Nevertheless, the definition of the state space is an important component in the formulation of a planning problem, and in the design and analysis of algorithms that solve it.

**Time:** All planning problems involve a sequence of decisions that must be applied over time. Time might be explicitly modeled, as in a problem such as driving a car as quickly as possible through an obstacle course. Alternatively, time may be implicit, by simply reflecting the fact that actions must follow in succession, as in the case of solving the Rubik’s cube. The particular time is unimportant, but the proper sequence must be maintained. Another example is a solution to the Piano Mover’s Problem; the solution to moving the piano may be converted into an animation over time, but the particular speed of motions is not specified in the planning problem. Just as in the case of state, time may be either discrete or continuous. In the latter case, we can imagine that a continuum of decisions are being made by a plan.

**Actions:** A plan generates actions that manipulate the state. The terms actions and operators are common in artificial intelligence; in control theory and robotics, the equivalent terms are inputs or controls. Somewhere in the planning formulation, it must be specified how the state changes when
1.3. Basic Ingredients of Planning

Figure 1.2: Remember puzzles like this? Imagine trying to solve one with an algorithm. The goal is to pull the two bars apart. This example is called the Alpha 1.0 Puzzle. It was created by Boris Yamrom, GE Corporate Research & Development Center, and posted as a research benchmark by Nancy Amato at Texas A&M University. This animation was made by James Kuffner, of Carnegie Mellon University. The solution was computed by the balanced, bidirectional RRT algorithm, developed by James Kuffner and Steve LaValle, and covered in Section 5.5.

Figure 1.3: Using robots to move a piano [1]. This solution was computed using planning techniques developed by Juan Cortés, Thierry Simeon, and Jean-Paul Laumond, and are covered in Section 7.4.
actions are applied. This may be expressed as an state-valued function for the case of discrete time, or as an ordinary differential equation for continuous time. For most motion planning problems, explicit reference to time is avoided by designing paths through a continuous state space. Such paths may be expressed as the integral of differential equations, but it is an unnecessary complication in this case. For some problems uncontrollable actions could be chosen by nature, which interfere with the outcome, but are not specified as part of the plan. This enables various forms of uncertainty to be introduced into the planning problem.

Initial and goal states: Planning generally involves starting in some initial state and trying to arrive at a specified goal state. The actions are selected in a way that causes this to happen.

A criterion: This encodes the desired outcome in terms of state and actions that are executed. There are generally two different kinds of planning concerns based on the type of criterion:

1. Feasibility: In this case, the only concern is whether the plan results in arriving at a goal state.

2. Optimality: Find feasible plans that optimize performance in some carefully specified manner, in addition to arriving in a goal state.

For most of the problems considered in this book, feasibility is already challenging enough; achieving optimality is considerably harder for most problems. Therefore, a substantial amount of focus is on finding feasible solutions to problems, as opposed to optimal solutions. The majority of literature in robotics, control theory, and related fields focuses on optimality, but this is not necessarily important for many problems of interest. In many applications, it is difficult to even formulate the right criterion to optimize. Even if a desirable criterion can be formulated, it may be impossible to obtain a practical algorithm that computes optimal plans. In such cases, feasible solutions are certainly preferable to having no solutions at all. Fortunately, for many algorithms, such as those developed in motion planning, the solutions produced are usually not too far from optimal in practice. This reduces the amount of motivation for finding optimal solutions. For problems that involve probabilistic uncertainty, however, optimization arises more frequently. The probabilities are often utilized to obtain the best performance in terms of expected costs. Feasibility is usually associated with performing worst-case analysis of uncertainties.

A plan: In general, a plan will impose a specific strategy or behavior on decision makers. A plan might simply specify a sequence of actions to be taken; however, it may be more complicated. If it is impossible to predict future states, the plan may provide actions as a function of state. In this case, regardless of future states, the appropriate action is determined. Using terminology from other fields, this enables feedback or reactive plans. It might even be the case that the state cannot be measured. In this case, the action must be chosen based on whatever information is available up to the current time. This will generally be referred to as an information state, on which a plan will be conditioned.

1.4 What is a Planning Algorithm?

![Turing Machine Diagram](Image)

Figure 1.4: According to the Church-Turing thesis, the notion of an algorithm is equivalent to the notion of a Turing machine.

What is a planning algorithm? This is a difficult question, which is difficult to completely answer in this section without formally introducing the planning concepts that appear in later chapters. One point needs to be made clear at this point: the classical Turing machine model used to define an algorithm in theoretical computer science is insufficient to encompass planning algorithms. A Turing machine is a finite state machine with a special head that can read and write along an infinite piece of tape, as depicted in Figure 1.4. The Church-Turing thesis states that an algorithm is a Turing machine (see [2, 6] for more details). The input to the algorithm is encoded as a string of symbols, usually a binary string, and then is written to the tape. The Turing machine reads the string, performs computations and then decides whether to accept or reject the string. This version of the Turing machine only solves decision problems; however, there are straightforward extensions that can yield other desired outputs, such as a plan.

The trouble with using the Turing machine as a model for planning algorithms is that plans will be generally assumed to interact with a physical world, as depicted in Figure 1.5. This is fundamental to robotics and many other fields in which planning is used. Using the Turing machine as a foundation for algorithms usually implies that the physical world must be first carefully modeled and written on the tape before the algorithm can make decisions. If changes occur in the world during execution of the algorithm, then it is not clear what should happen. For example, a mobile robot could be moving in a cluttered environment in which people are walking around. The robot might throw an object onto a table without being able to precisely predict how the object will come to rest. It can take measurements of the results with sensors, but it again becomes a difficult
1.4. WHAT IS A PLANNING ALGORITHM?

Figure 1.5: a) The boundary between machine and environment is considered as an arbitrary line that may be drawn in many ways depending on the context. b) Once the boundary has been drawn, it is assumed that the machine interacts with the environment through sensing and actuation.

The task to determine how much should be explicitly modeled and written on the tape. The on-line algorithm model is more appropriate for these kind of problems [3, 4, 7]; however, it still does not capture a notion of planning algorithms that is sufficiently broad for the topics of this book.

Figure 1.6: A robot could be used to simulate a Turing machine. Through manipulation, many other kinds of behavior could be obtained that fall outside of the Turing model.

The processes that can occur in a physical world are more complicated than the interaction between a state machine and a piece of tape filled with symbols. It is even possible to simulate the tape by imagining a robot that interacts with a long row of switches as depicted in Figure 1.6. The switches serve the same purpose as the tape, and the robot carries a computer that can simulate the finite state machine.\(^1\) The complicated interaction allowed between a robot and its

\(^1\)Of course, simulating infinitely-long tape seems impossible in the physical world. Other versions of Turing machines exist in which the tape is finite, but unbounded. This may be more appropriate for the current discussion.

Figure 1.7: In a hierarchical model, the environment of one machine may itself contain a machine.

Figure 1.8: A hierarchical model that has been used for decades in robotics.

Environment could give rise to many other models of computation. A discussion of performing computations with mechanical systems is given in [5].

In general, the physical world will be referred to as the environment. The device that implements a plan will be referred to as the machine. Practical examples of the machine include a robot, a piece of software, or even specialized hardware which may be digital or analog. As indicated in Figure 1.5, the boundary between the machine and the environment is an arbitrary line that varies from problem to problem. Once drawn, sensors provide information about the environment which serves as input to the machine during execution. The machine then executes actions, which provides actuation to the environment. The actuation may alter the environment in some way that is later measured by sensors. Therefore, there is close coupling between the machine and its environment during execution.

It is even possible to draw the line between machine and environment in multiple places, which results in a hierarchical model. The environment, \(E_1\), with respect to a machine, \(M_1\), might actually include another machine \(M_2\) that interacts with its environment, \(E_2\), as depicted in Figure 1.7. Figure 1.8 shows a typical hierarchy used for years in robotics. In general, any number of planning layers may be defined. For the design of planning algorithms, reference will usually only be made to a single layer. If the models are formulated correctly for each layer, and if each designed plan functions correctly, then the global hierarchy should solve tasks as desired. There are many interesting issues involving the construction of such hierarchies, but these will not be addressed in this book because they depend heavily on the particular context in which planning is used. Determining the appropriate places to draw boundaries and modularize a complicated problem is mostly the burden of the expert who applies planning techniques in a particular context.

Once the boundary has been drawn between the machine and its environment, a third component can be introduced: the planner. The task of the planner is to produce a plan based on a description of possible environments. There are
two general models for plans constructed by the planner. The first case is depicted in Figure 1.9, in which the planner produces a plan, which is encoded in some way and given as input to the machine. In this case, the machine is considered programmable, and can accept possible plans from planner before execution. It will generally be assumed that once the plan is given, the machine becomes autonomous and can no longer interact with the planner.²

The second general model for plans is depicted in Figure 1.10. In this case, the plan produced by the planner encodes an entire machine. The plan can be considered as a special-purpose machine that is designed to solve the specific tasks given originally to the planner. Under this interpretation, it may be preferable to be minimalist and design the simplest machine possible that sufficiently solves the desired tasks.

There are two possible ways to implement the planner. The planner may either be an algorithm in the Turing machine sense (or some related variant), or the planner may even be a human. For example, it is perfectly acceptable for a human to design a state machine that is connected to the environment. There are no additional inputs in this case because the human fulfilled the role of the traditional algorithm. The environment model is given as input to the human, and the human “computes” a plan. An example in which the planner is a traditional algorithm is given in robotics by classical motion planning. An algorithm receives a description of the environment in terms of geometric models and then computes a plan, which is a collision-free path to be followed by the robot. Whether the planner is a human or is a machine itself, the process of developing plans will be generally referred to as planning algorithms.

To summarize, there are three general components:

1. The environment, which models the physical world with which a plan must interact.
2. The machine, which interacts with the environment through sensing and actuation. The machine may be programmable, which means a plan can be downloaded, or the machine may simply be the plan itself.
3. The planner, which takes one of a set of environment descriptions and produces a plan. In some cases, the human designs the planner, and in others, the human is the planner. In both cases, these will be referred to as planning algorithms.

1.5 Organization of the Book

PART I: Introductory Material

²Of course this model can be extended to allow machines to be improved over time by receiving better plans; however, we want a strict notion of autonomy for the discussion of planning in this book. This model does not prohibit the updating of plans in practice.
This provides very basic background for the rest of the book.

- **Chapter 1: Introductory Material**
  This includes some examples and provides a high-level overview of planning philosophy.

- **Chapter 2: Discrete Planning**
  This chapter can be considered as a springboard for entering into the rest of the book. From here, you can continue to Part II, or even head straight to Part III. Sections 2.1 and 2.2 are most important for heading into Part II. For Part III, Section 2.3 is additionally useful.

**PART II: Motion Planning**
The main source of inspiration for the problems and algorithms covered in this part comes from robotics. The methods, however, are general enough to apply to applications in other areas, such as computational biology, computer-aided design, and computer graphics. An alternative title that more appropriately reflects the kind of planning that occurs is “Planning in Continuous State Spaces.”

- **Chapter 3: Geometric Representations and Transformations**
The chapter gives important background for expressing a motion planning problem. Section 3.1 describes how to construct geometric models, and the remaining sections indicate how to transform them. Sections 3.1 and 3.2 are most important for later chapters.

- **Chapter 4: The Configuration Space**
  This chapter introduces concepts from topology and uses them to formulate the configuration space, which is the state space that arises in motion planning. Sections 4.1, 4.2, and 4.3.1 are most critical for understanding most of the material in later chapters. In addition to the previously mentioned sections, all of Section 4.3 provides useful background for the combinatorial methods of Chapter 6.

- **Chapter 5: Sampling-Based Motion Planning**
  This chapter introduces motion planning algorithms that have dominated the literature in recent years and have been applied in many applications both in and out of robotics. If you understand the basic idea that the configuration space represents a continuous state space, most of the concepts should be understandable. They even apply to other problems in which continuous state spaces emerge, in addition to motion planning and robotics.

- **Chapter 6: Combinatorial Motion Planning**
  The algorithms covered in this section are sometimes called exact algorithms. They provide complete (i.e., they find a solution if one exists, or report failure, otherwise) solutions to motion planning problems. The sampling-based algorithms have been more useful in practice, but these are not complete in the same sense.

- **Chapter 7: Extensions of Basic Motion Planning**
  This chapter introduces many problems and algorithms that are extensions of the methods from Chapters 5 and 6. Most can be followed with basic understanding of the material from these chapters. Section 7.4 covers planning for closed kinematic chains; this requires an understanding of the additional material, which is covered in Section 4.4.

- **Chapter 8: Feedback Motion Planning**
  This is a transitional chapter that introduces feedback into the motion planning problem, but still does not introduce differential constraints, which is deferred until Part IV. The previous chapters of Part II focused on computing open loop plans, which means that any errors that might occur during execution of the plan are ignored, yet the plan will be executed as planned. Using feedback enables the plan to respond to execution errors.

**PART III: Decision-Theoretic Planning**
An alternative title is “Planning under Uncertainty”. Most of the part addresses discrete state spaces, which can be studied immediately following Part I. However, some sections cover extensions to continuous spaces; to understand these parts, it will be helpful to have read some of Part II.

- **Chapter 9: Basic Decision Theory**
  The concepts and concepts developed here involve making decisions in a single step, but in the face of uncertainty. Therefore, the problems generally are not considered planning, and there is no talk of state spaces. This chapter provides important background for Part III, however, because planning under uncertainty can be considered as multi-step decision making. Chapter 9 covers a single step, which is used as a building block for later planning concepts.

- **Chapter 10: Sequential Decision Theory**
  This chapter takes the concepts from Chapter 9 and extends them by chaining together a sequence of basic decision-making problems. Dynamic programming concepts from Section 2.3 become important here. For all of the problems in this chapter, it is assumed that the current state is always known. All uncertainties that exist are with respect to prediction of future states, as opposed to measuring the current state.

- **Chapter 11: The Information Space**
  The chapter defines a framework for planning when the current state is not known. Information regarding the state is obtained from sensor observations and memory of actions that were previously applied. The information
space serves a similar purpose for problems with sensing uncertainty as the configuration space did for motion planning.

- **Chapter 12: Planning in the Information Space**
  This chapter covers several planning problems and algorithms that involve sensing uncertainty.

**PART IV: Planning under Differential Constraints**
This can be considered as a continuation of Part II. Now there can be both global (obstacles) and local (differential) constraints on the continuous state spaces that arise in motion planning. Dynamical systems are also considered, which yields state spaces that include both position and velocity information (this coincides with the notion of a *state space* in control theory or a *phase space* in physics and differential equations).

- **Chapter 13: Differential Models**
  This chapter serves as an introduction to Part IV by giving examples of differential constraints that arise in practice and explaining how to model them in the context of planning.

- **Chapter 14: Sampling-Based Planning Under Differential Constraints**
  Several sampling-based algorithms for planning under differential constraints are covered. Many are extensions of methods from Chapter 5. There is no chapter on combinatorial methods because very little can be accomplished with combinatorial techniques in the context of differential constraints.

- **Chapter 15: System Theory and Analytical Techniques**
  This chapter provides an overview of important theory developed for the control of nonlinear systems. The concepts and tools provide important background that can be used to understand and improve planning algorithms that handle differential constraints.
Bibliography


