Automatic Generation of Four-part Harmony

Liangrong Yi
Computer Science Department
University of Kentucky
Lexington, KY 40506-0046

Judy Goldsmith
Computer Science Department
University of Kentucky
Lexington, KY 40506-0046

Abstract

This paper introduces decision-theoretic planning techniques into automatic music generation. Markov decision processes (MDPs) are a mathematical model of planning under uncertainty, and factored MDPs demonstrate great advantages in an environment where the state space is large. Given a melody, we use a factored MDP planner to fill in the other three voices, based on the characteristics of classical Western four-part harmony.

1 INTRODUCTION

Our research group is interested in decision-theoretic planning algorithms. We believe that our nondeterministic planning research can also produce good applications in music generation. For the four-part harmony problem, the process of adding the other three voices when given the melody can be viewed as a planning problem. We create an automatic harmony generator based on our factored MDP planner.

2 FUNDAMENTALS OF HARMONY

2.1 WHAT IS HARMONY?

The music of most ancient cultures is monophonic, which means it consists of individual melodies without accompaniment. Harmony was developed early in the Middle Ages. Harmony is a group of notes which sound at the same time. The use of harmony makes music rich and colorful. In our research, we only consider one type of chord: the triad. A triad is a three-note chord built in thirds [Ottman, 1970]. Conventionally, in English, pitches are named with the first seven letters of the alphabet, A B C D E F G. Figure 1 shows an example triad.

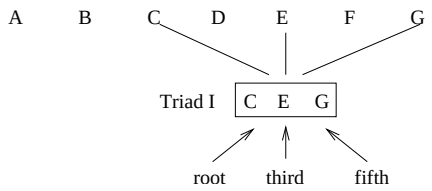

Figure 1: An Example Triad

A triad can appear in three different positions. In a root position triad, the lowest note is known as the **root**, above which are the **third** and **fifth**. If the third or fifth of the triad is the lowest note, then the triad is in **inversion**. A triad is in **first inversion** if the third is the lowest note; a triad is in **second inversion** if the third is the lowest note. Figure 2 (see also in [Ottman, 1970]) illustrates the triads in the Major Scale (C Major). Major triads are represented by Roman numerals in uppercase and minor triads are represented by Roman numerals in lowercase.

Scale Degree Name	Triad Number	Spelling in C Major
Tonic	I	C E G
Supertonic	ii	D F A
Mediant	iii	E G B
Subdominant	IV	F A C
Dominant	V	G B D
Submediant	vi	A C E
Leading Tone	vii^o	B D F

Figure 2: Triads in C Major

The elementary study of harmony is about chord progression. A chord progression is a series of chords

played in order. When writing harmony, we need to consider the connection between chords. Much attention is placed on cadences. The cadence, in music, is the place which gives the feeling of a temporary stop or a full stop [Ottman, 1970]. The cadence often consists of two successive chords. The two types of cadences are authentic cadences, which use tonic and dominant chords (V-I, I-V, v-i, i-v) and plagal cadences, which use tonic and subdominant chords (IV-I, I-IV, iv-i, i-iv).

Our research is now focused on four-part harmony, also known as four-voice texture. One of the most familiar examples is choir music (soprano, alto, tenor, bass). In four-part harmony, we need to consider the appropriate range of each voice, chordal spacing among four voices, voice movement within the same triad and doubling in four-voice chords [Gauldlin, 1996]. Given the melody line, the goal is to automatically generate the other three parts. Of course, we hope that the music will follow the classical western harmony theory in principle and be acceptable to musicians and audience.

2.2 MUSIC NOTATION

The **abc** notation [Walshaw, 2007] provides computer researchers a good way to represent music in **ASCII** format. Originally it was designed to record folk and traditional Western European tunes. But it is extensible to other types of music. In abc notation, the Middle C is notated as C, the D immediately above middle C is notated as D, and so on up the scale. The next C in the higher octave is notated in lowercase as c. The next octave up is shown by an apostrophe immediately after the note name, like c′. The octave immediately below middle C is represented by a comma immediately following the note name, e.g. B,. Here is the four octaves in abc notation:

$$C, D, E, F, G, A, B, CDEFGABcdefgabc'd'e'f'g'a'b'$$

The users can further extend the notation by adding more commas or apostrophes.

The abc notated music can be easily read by both computers and humans. With a little training, musicians can play a tune directly from the abc notation without having to change it to sheet music. There are many software packages which can read and process abc notations. Some tools can even play music in abc notation. All this make abc notation a good choice for our research. We can easily write the inputs of the planner, compare the results, and play the resultant music or let the computer play it.

3 EXISTING WORK ON HARMONY GENERATION

Horner and Ayres [Horner and Ayers, 1995] made some early efforts to generate four-part harmony using **genetic algorithms** (GAs). Their work was very successful, but on a very constrained problem (chords were given) [Biles, 2007]. Somnuk Phon-Amnuaisuk *et al.* also worked in this area [Phon-Amnuaisuk et al., 1999, Phon-Amnuaisuk and Wiggins, 1999]. In their research work, the soprano information is input by the users. The GA generates the other three voices with musical domain knowledge encoded in the fitness function.

Research has been done to use constraints to model musical harmonization. Francois Pachet and Pierre Roy surveyed this area [Pachet and Roy, 2001].

In [Allan and Williams, 2005], a **hidden Markov model** (HMM) is used to model the harmony progression. A hidden Markov model is a Markov process with hidden states. In this harmony model, the visible states represent melody notes and the hidden states are chords.

Dan Ponsford *et al.* [Ponsford et al., 1999] developed a grammar-learning system to learn significant characteristics of the music from a corpus of music examples. That grammar can then be used to generate new harmonies in the same style.

4 MARKOV DECISION PROCESSES

Markov decision processes (MDPs) are considered a good way to model stochastic systems. Chord progression can be viewed as a stochastic process. The choice of a chord is similar to the choice of an action in MDP planning. If we use utilities to decide the goodness of the harmony, then we will want to pick a set of chords which can maximize the utilities. Therefore, we believe that we can use MDP based planning techniques to automate harmony generation.

An (MDP) is a five-tuple $\langle S, A, Pr, R, C\rangle$. S is a finite state space, which contains all possible states. A state s is a description of the system at a particular time. A is a finite action space. At each time point t of the process and each state s, the agent has a set of available actions. After the agent takes an action, the system may change from one state to another. The state transition function Pr describes how the next state depends on the current state and action. Pr is a mapping from $S \times A \times S$ into a real number in $[0, 1]$, so that $Pr(j|i, a)$ defines the probability that the system moves from state i to state j by taking the action a.

The function $R : S \rightarrow \mathbf{R}$ assigns a reward to a state s. The cost function $C : S \times A \rightarrow \mathbf{R}$ associates a cost with performing an action a in state s. The combination of the reward function and the cost function gives the utility of each state. A policy is a mapping from a state to an action. In MDP planning, the goal is to find an optimal policy which maximizes the value of each state.

There are two basic formats to represent MDPs. One is the explicit or extensional representation in which states are enumerated directly. The other is factored or intensional representation. What we will use in harmony generation is the factored one.

Instead of enumerating states explicitly, a factored representation defines a set of attributes that are sufficient to describe the states [Boutilier et al., 1999]. Compared to explicit representations, factored representations are considered more convenient and compact in many situations.

A factored MDP is described by a set of state variables $\mathbf{X} = \{\mathbf{X_1}, ..., \mathbf{X_n}\}$. Values of each variable are denoted in lowercase (e.g., x_i). Each X_i takes on values in some finite domain $Dom(X_i)$. A state x defines a value $x_i \in Dom(X_i)$ for each variable X_i [Guestrin et al., 2003].

In factored MDPs, transition functions are usually described by **dynamic Bayesian networks** (DBNs). A **Bayesian network** [Pearl, 1988] is a representational framework for compactly representing a probability distribution in factored form. Formally, a Bayesian network is a directed acyclic graph in which vertices correspond to random variables and an edge between two variables indicates a direct probabilistic dependency between them [Boutilier et al., 1999]. Each variable is associated with a **conditional probability table** (CPT) which specifies its probability conditioned on all possible values of its immediate parents in the graph. A **two-stage temporal Bayes net** (2TBN) [Boutilier et al., 1999] is commonly used to describe the state-transition probabilities of an action in an MDP.

A **binary decision diagram** (BDD) is a directed acyclic graph with a single root [Andersen, 1997]. A non-terminal vertex in a BDD is labeled with a variable and has two children. One of the edges from the non-terminal vertex v to one of its children is labeled with **TRUE**; the other edge is labeled with **FALSE**. A terminal vertex (leaf) is labeled with a Boolean value and has no outgoing edges. A BDD maps n Boolean variables to a Boolean value, which is specified by the terminal vertex. **Algebraic decision diagrams** (ADDs) generalize BDDs. In ADDs, a terminal vertex can take real number values.

In factored MDP planning, we usually use ADDs to represent states. ADDs can represent state transition probabilities as well. With the ability to capture regularities in the CPTs, ADDs fully take advantages of the efficiency of factored MDPs. We can also use ADDs to represent policies. In a **policy ADD**, the non-terminal vertices are state variables and the terminal vertices are labeled with actions. Therefore a policy ADD maps a set of states to a set of actions.

Jesse Hoey *et al.* [Hoey et al., 1999] proposed **stochastic planning using decision diagrams** (SPUDD). SPUDD is an algorithm for finding optimal or near-optimal policies for factored MDPs. The algorithm is based on value iteration, but it uses ADDs to represent value functions and CPTs. We built a planner based on SPUDD, with a few modifications according to our specific domain.

5 MODEL AND RESULTS

5.1 FROM HARMONY TO MDP

To simplify the problem, we only take care of the melodic notes at each beat and only consider adding triads at each beat. In the beginning of our work, all the examples are in 4/4 rhythm and C major.

At each beat, there are four notes, one from each of the four different parts. They can be considered as variables of a state. One is known (the melody), the other three are what we are trying to generate. Since we add chords at each beat, we do not need to worry about duration.

In our model, a state is made up of ten variables $\mathbf{S_1}$, $\mathbf{A_1}$, $\mathbf{T_1}$, $\mathbf{B_1}$, $\mathbf{S_2}$, $\mathbf{A_2}$, $\mathbf{T_2}$, $\mathbf{B_2}$, $\mathbf{S_3}$, $\mathbf{P}$. The first nine variables correspond to music notes, represented in abc notation. To make it legible, we use $\mathbf{S}$, $\mathbf{A}$, $\mathbf{T}$, $\mathbf{B}$ to distinguish the notes from soprano, alto, tenor and bass respectively. The subscript tells whether the note comes from the first chord or the second chord. $\mathbf{P}$ records the position of the first chord in this state. An example is shown in Figure 3.

In the hope of having a manageably small MDP, we focus on two successive chords. That is the reason why we include two chords in a state. By doing this, we can tell how good the connection of the two chords is by computing the utilities of that state [1].

Actions are defined as the choice of a specific chord. The focused attention on triads leads to a fairly small action space. We use 7 actions (7 triads) in the major scale or 13 actions in the minor scale. At each

[1] Most rules of harmony consider longer sequences of chords. We will take into account longer sequences as soon as we have planners that can handle such models.

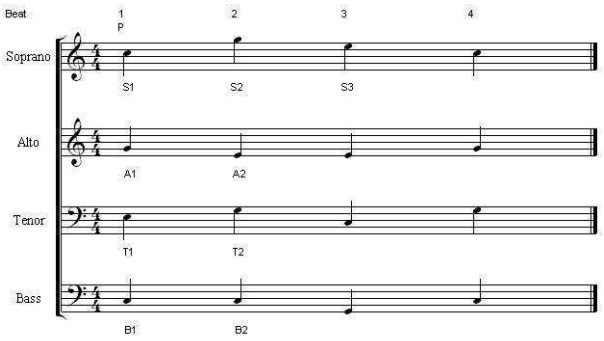

Figure 3: An Example State in the Harmony Model

state, even if the triad for the next beat is decided, uncertainty still exists. A triad can appear in the root position, the first inversion or the second inversion. Since we only generate triads and the music is made up of four parts, there might be repeats for notes (eg. CEGc). It is also possible to omit a note (eg. CGcc). Those probabilities can be either defined in advance or learned from a corpus of music.

At each state, to choose an action, utility needs to be computed. A utility function can be defined according to classical harmony theory or learned from a specific collection of music if we want to generate harmony in that musical style. One way of defining the utility function is to give a penalty when breaking a rule. That can be done by giving negative rewards.

We start our experiments with music in C Major. There are seven commonly used triads for a major scale, which are already shown in Figure 2. Those triads are numbered by their corresponding triad numbers. As long as an action is picked, the values of the variables $\mathbf{A_2}$, $\mathbf{T_2}$, $\mathbf{B_2}$ in the next state are limited to the notes in the chosen triad.

The DBN of this model is shown in Figure 4. Actually Figure 4 is the DBN for action **I**, i.e. choosing triad **I** (CEG). The CPT for variable $\mathbf{A'_2}$ is also shown in the figure and represented by a decision diagram. According to our definition of a state, one chord may appear in two states. That results in redundancy. We can see that the values of $\mathbf{S'_1}$, $\mathbf{A'_1}$, $\mathbf{T'_1}$, $\mathbf{B'_1}$, $\mathbf{S'_2}$ are the same as $\mathbf{S_2}$, $\mathbf{A_2}$, $\mathbf{T_2}$, $\mathbf{B_2}$, $\mathbf{S_3}$. Therefore, the transitions from $\mathbf{S_2}$ to $\mathbf{S'_1}$, $\mathbf{A_2}$ to $\mathbf{A'_1}$, $\mathbf{T_2}$ to $\mathbf{T'_1}$, $\mathbf{B_2}$ to $\mathbf{B'_1}$ and $\mathbf{S_3}$ to $\mathbf{S'_2}$ are deterministic.

We use a vector ($\mathbf{s_1}$ $\mathbf{a_1}$ $\mathbf{t_1}$ $\mathbf{b_1}$ $\mathbf{s_2}$ $\mathbf{a_2}$ $\mathbf{t_2}$ $\mathbf{b_2}$ $\mathbf{s_3}$ $\mathbf{p}$) to represent a state. For example, the current state in Figure 3 is (c G E, C, g E G, C, e 1). Now we pick action **I** (triad I), then the next state is ($\mathbf{s'_1}$ $\mathbf{a'_1}$ $\mathbf{t'_1}$ $\mathbf{b'_1}$ $\mathbf{s'_2}$ $\mathbf{a'_2}$ $\mathbf{t'_2}$ $\mathbf{b'_2}$ $\mathbf{s'_3}$ $\mathbf{p'}$). We know that $\mathbf{s'_1} = \mathbf{s_2}$, $\mathbf{a'_1} = \mathbf{a_2}$, $\mathbf{t'_1} = \mathbf{t_2}$, $\mathbf{b'_1} = \mathbf{b_2}$, $\mathbf{s'_2} = \mathbf{s_3}$. $\mathbf{s'_3}$ will be given

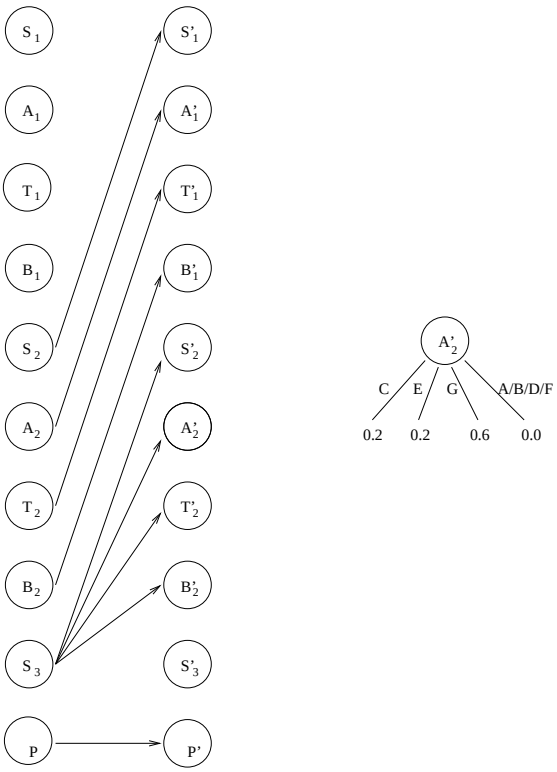

Figure 4: The DBN for Action **I** (choosing triad I: CEG)

by the melody. $\mathbf{p'}$=2, since the first triad of this state is at the second beat of the whole music. The values of $\mathbf{A'_2}$,$\mathbf{T'_2}$,$\mathbf{B'_2}$ are restricted to C, E, or G (here we ignore the different scales). Uncertainty still exists. The chord can appear:

- in root position: $\mathbf{a'_2}$ = **G** $\mathbf{t'_2}$=E $\mathbf{b'_2}$=C
- in first inversion: $\mathbf{a'_2}$=C $\mathbf{t'_2}$=G $\mathbf{b'_2}$=E
- in second inversion: $\mathbf{a'_2}$=E $\mathbf{t'_2}$=C $\mathbf{b'_2}$=G
- with doubling: $\mathbf{a'_2}$=C $\mathbf{t'_2}$=E $\mathbf{b'_2}$=C
- or other combinations.

Our model is a finite MDP. The horizon of this MDP is the total number of beats given by the melody. In planning with finite horizon, the i^{th} iteration in the value iteration algorithm represents the value function for the states which are i steps away from the end of the execution. In the SPUDD algorithm, to get the optimal value function, the value iteration proceeds to construct a series of **n-step to go** value functions $\mathbf{V^n}$.

$$V^n(s) = R(s) + \max_a [\gamma \sum_{s' \in S} Pr(s'|s,a)) V^{n-1}(s')]$$

At present, we only consider two rules:

1. The melodic note is better to appear in the corresponding chord.
2. We prefer authentic cadences, but we also accept plagal cadences.

The two rules are encoded into the utility functions.

5.2 EXPERIMENTS

We ran our harmony generator on Beethoven's **Ode to Joy**. The music was divided into small units. Each unit consisted of two measures, i.e. 8 beats. On each unit, we picked the first two chords (the initial state) and then implemented finite horizon planning. Given the input of the melody in abc notation, the factored MDP planner produced an optimal policy, which was a series of actions. Each action corresponded to a triad at a specific beat.

Although we get the chord progression, we still need to generate the exact notes for the three parts. We created a program to simulate the execution of the policy. The simulator took the action according to the optimal policy. As we discussed in Section 5.1, actions are described by DBNs. $\mathbf{s'_1}, \mathbf{a'_1}, \mathbf{t'_1}, \mathbf{b'_1}, \mathbf{s'_2}$ in the next state would be the same as $\mathbf{s_2}, \mathbf{a_2}, \mathbf{t_2}, \mathbf{b_2}, \mathbf{s_3}$ in the current state. $\mathbf{s'_3}$ is given by the melody. $\mathbf{p'}$ increases one at each step. After taking the action, the values for $\mathbf{a'_2}, \mathbf{t'_2}, \mathbf{b'_2}$ are decided by the CPTs of the respective variables. The CPTs give us the probabilities of the values. In each simulation, the program randomly chooses the next state according to the transition probabilities. If we run the simulator multiple times, we may get **a different harmony each time**. The harmonies are made up of the same series of chords, but the individual voices will be different. Those different harmonies are considered different views of the policy. Human users have the option of choosing the harmony they prefer. We picked two from the experiments of **Ode to Joy**, shown in Figure 5 and Figure 6. Although the chords progressions are the same, the two pieces are slightly different. The first one in Figure 5 has more repeats, while the second one in Figure 6 encounters some big jumps in an individual voice.

The potential users of the harmony generation tool would be amateur music lovers. Sometimes they want to add harmonies to the music they like or to the melody composed by themselves. But they probably do not have enough music theory knowledge to do so. Music can be translated into abc notation by hand or by special tools. With the input in abc notation, the automatic harmony generator will output a number of four-part harmony pieces with the same melody.

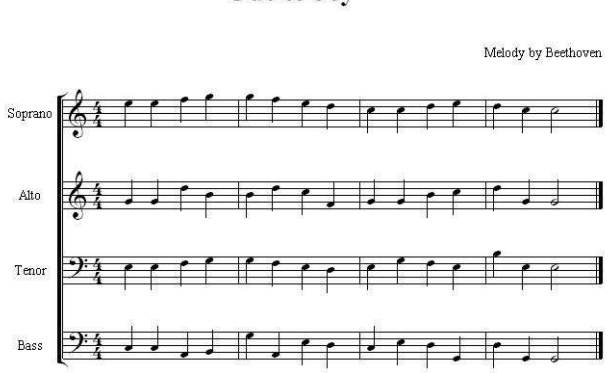

Figure 5: Generated Harmony 1

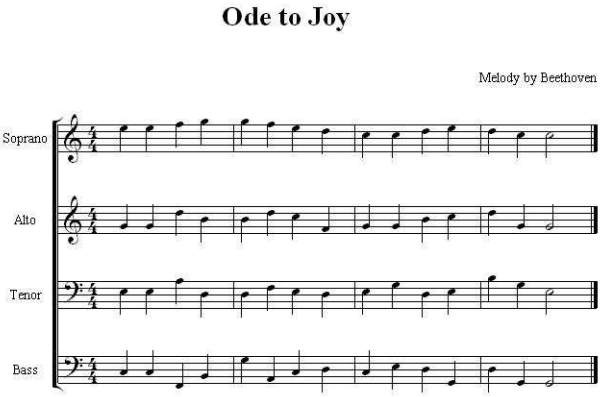

Figure 6: Generated Harmony 2

The users then hear those pieces through the speakers of the computers or play the pieces themselves. They can pick whichever they like best. Of course, they may still want to make some changes to the selected one. But that would be easier than writing harmonies from scratch.

6 Conclusions

We introduced the idea of using factored MDPs to model harmony generation. One specialty of the model is that the values of some variables ($\mathbf{S_1}$, $\mathbf{S_2}$ $\mathbf{S_3}$) are fixed before planning. That is because the melodic notes are already given. Except for that, the planning process is almost the same as the general MDP planning. From the perspective of musicians, the input of

the melody will result in the output of the four-part harmony in abc notation. The common rules used to write harmonies are encoded into the utility functions of the MDP model. The harmony generator helps them to do the tedious analyzing and writing process.

Our work shows that it is possible to apply decision-theoretic planning techniques to automate music generation. Given a melody, our four-part harmony generator can produce reasonable music. The current harmonies that are produced are not very sophisticated, but we are confident that improved utility functions will improve the quality of the generated harmony. For instance, in the example of **Ode to Joy**, there are several places where the same note is played by two adjacent parts, which leads to a less full sound. We also encounter big jumps within a part, which leads to more difficult individual parts.

In classical four-part harmony, many factors are taken into consideration. Our current model is preliminary, and only includes two rules. As we integrate more rules into the model, we believe that the generated music will be much better. We will be investigating the effects of different rules in the near future.

Acknowledgements

This work is partially supported by NSF GRANT ITR-0325063.

References

Robert Ottman. *Elementary harmony*. Prentice-Hall, 1970.

Robert Gauldlin. *Harmonic practice in tonal music*. W.W.Norton & Company, 1996.

Chris Walshaw. abc music notation, 2007. http://www.walshaw.plus.com/abc/.

Andrew Horner and Lydia Ayers. Harmonisation of musical progression with genetic algorithms. In *proceedings of the 1995 International Computer Music Conference (ICMC-95)*, 1995.

Al Biles. Evolutionary music bibliography, 2007. http://www.it.rit.edu/∼jab/EvoMusic/EvoMusBib.html.

Somnuk Phon-Amnuaisuk, Andrew Tuson, and Geraint Wiggins. Evolving musical harmonisation. In *proceedings of the Fourth International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA-99)*, 1999.

Somnuk Phon-Amnuaisuk and Geraint Wiggins. The four part harmonisation problem: A comparison between genetic algorithms and a rule-based system. In *Proceedings of the AISB'99 Symposium on Musical Creativity*, 1999.

Francois Pachet and Pierre Roy. Musical harmonization with constraints: A survey. *Constraints Journal*, 6(1):7–19, 2001.

Moray Allan and Christopher K.I. Williams. Harmonising chorales by probabilistic inference. *Advances in Neural Information Processing Systems*, 17, 2005.

Dan Ponsford, Geraint Wiggins, and Chris Mellish. Statistical learning of harmonic movement. *Journal of New Music Research*, 28(2):150–177, 1999.

Craig Boutilier, Thomas Dean, and Steve Hanks. Decision-theoretic planning: structural assumptions and computational leverage. *Journal of AI Research*, 11:1–94, 1999.

Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research*, 19:399–468, 2003.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, 1988.

Henrik Reif Andersen. An introduction to binary decision diagrams, 1997. Lecture notes for 49285 Advanced Algorithms E97.

Jesse Hoey, Robert St-Aubin, Alan Hu, and Craig Boutilier. SPUDD: Stochastic planning using decision diagrams. In *the Fifteenth Conference on Uncertainty in Articial Intelligence (UAI-99)*, 1999.