PARTIALLY OBSERVABLE MARKOV DECISION PROCESSES FOR INTERACTIVE MUSIC SYSTEMS

Aengus Martin, Craig Jin, André van Schaik

School of Electrical and Information Engineering,
Sydney University, NSW 2006, Australia

William L. Martens

Faculty of Architecture, Design and Planning,
Sydney University, NSW 2006, Australia

ABSTRACT

Partially observable Markov decision processes provide a mathematical framework for interaction under uncertain conditions. In this paper we demonstrate their use as a convenient mechanism to control the behaviour of an interactive music system, where otherwise it might be required to use ad-hoc heuristic schemes which are difficult to manage and fine-tune.

1. INTRODUCTION

In artificial intelligence, an agent is "anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors" and a rational agent is one which acts to achieve certain goals [9]. An interactive music system [8] (IMS) designed to improvise alongside a musician is an example of a rational agent. It must act to achieve its musical goals, as set by its designer. This can mean making choices related to style, harmony, rhythm or any musical attribute. Such choices should use knowledge of (i) the behaviour of the musician and (ii) his likely responses. The partially observable Markov decision process[1] (POMDP) is a mathematical model of the interaction between an agent and its environment. It provides a mechanism by which the agent can be programmed to act optimally with respect to a set of goals. This paper reports on an investigation into the use of POMDPs in IMSs designed to improvise alongside a musician. To our knowledge, POMDPs have not previously been applied in this area.

The designers of IMSs do not always publish the precise mechanisms by which musical choices are made. Many are based on complex combinations of heuristic and stochastic processes, such as Lewis' *Voyager* [7]. Collins [1] describes a number of systems including *Free improvisation simulation* which involves improvising agents with human-like characteristics such as 'keenness', 'shyness', and 'insularity'. In the following, we will show that the POMDP is a convenient mechanism for designing and tuning behavioural traits such as these, where otherwise ad-hoc combinations of heuristics might be required.

Models related to the POMDP include hidden Markov models (HMMs) and Markov decision processes (MDPs). HMMs can also be used to take advantage of knowledge of musical behaviour. They have been used in this way to improve the identification of chords and keys in recorded music [6]. However HMMs cannot account for the effects that an agent's actions might have on a musician and they cannot be used to specify musical goals. MDPs have been used as the basis of a four-part harmony generator [12] but we have found no reference in the literature to their use in interactive music applications.

The structure of this paper is as follows. First we describe a simple IMS which plays music along with an improvising musician. The system uses information about the musician's performance to choose the key in which it will play. We use a POMDP model in the module which chooses the key. This module is described in detail in Section 3. In Section 4 we describe the software developed to allow POMDPs to be used in the Max[2] interactive platform. We then present some informal observations made while using our IMS. Finally we discuss the application of POMDPs to other interactive computer music scenarios and conclude.

2. THE TONAL IMPROVISING SYSTEM

We have implemented a simple interactive music system on the Max platform. We refer to it as the Tonal Improvising System (TIS, see Figure 1). It is designed for use by a musician playing a monophonic, MIDI-enabled instrument. He plays along with a metronome which sets the beat of the music. The TIS receives the MIDI note data associated with his performance. Every two bars, a key-finding module estimates the key in which the musician has been playing over the previous two bars. Then a key-choosing module uses this estimate to choose the key in which the TIS will play. It is in the key-choosing module that a POMDP is used. Finally, an improvisation-generating module takes the chosen key as input, and outputs a synthesized 'improvisation' in that key. This system will now be described in detail.

[1] The POMDP model originated in the area of operations research and there is a large body of literature on their application and solution. A good starting point is [4].

[2] www.cycling74.com

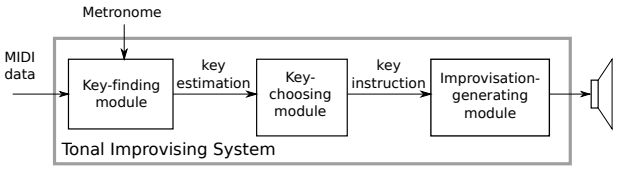

Figure 1. Schematic of the tonal improvising system.

The key-finding module is an implementation of the key-finding method due to Krumhansl [5]. Out of the large selection of algorithms available [11], we chose this method for its simplicity and ease of implementation in Max. Our implementation works as follows. The MIDI note data produced by the musician's performance is analysed in two-bar segments. For each segment, a 12-vector is calculated, indicating the length of time spent during the segment in each pitch class. The correlation between this vector and each of 24 key-profiles (for details, see [5]) is calculated. The key corresponding to the key-profile which gives rise to the highest correlation is deemed to be the key of the segment.

The key-choosing module chooses the key in which the TIS will play for the following two bars. It uses a POMDP model to define a mapping from the key estimate to the key in which the TIS should play. This mapping is governed by the parameters of the POMDP, which are set by the system designer. The POMDP model is described in Section 3. Each time the key is re-estimated, the key-choosing module outputs one of 25 possible instructions to the improvisation-generating module. There is one instruction of the form "play in key X" for each musical key and an additional instruction which indicates that no specific key should be played. We refer to this as the non-specific key instruction.

The improvisation-generating module takes instructions from the key-choosing module as input. It remains silent when it receives the non-specific key instruction and for the remaining instructions it plays the root note of the specified key once on each beat. This extremely simple generator is of little musical interest but was sufficient to experiment with the TIS. The output of the improvisation-generating module can be heard by the musician via a loudspeaker.

3. CHOOSING THE KEY IN WHICH TO PLAY

The POMDP is a discrete-time model of the interaction between an agent and its environment. The TIS is an agent and the 'environment' with which it interacts is the musician. It perceives the musician using the key-finding module and it acts upon the musician by controlling the key used by the improvisation-generating module. According to the POMDP model (see Figure 2), the environment exists at time t_i in a particular state, s_i. The agent makes an observation, o_i, at time t_i which provides some information about the current state of the environment. The agent uses the observation to choose an the action to take, a_i. This action has an effect on the environment so that later on, at time t_{i+1}, the environment has transitioned to a new state, s_{i+1}, which is probabilistically dependant on the action taken and the previous state. The agent then makes a new observation, o_{i+1} and chooses a new action, a_{i+1}, and so on. This process is governed by the parameters of the POMPD model which are denoted by ($\mathscr{S}$, $\mathscr{O}$, $\mathscr{A}$, $\mathfrak{T}$, $\mathfrak{Z}$, $\mathfrak{R}$, γ, b_0). These quantities are defined in the following.

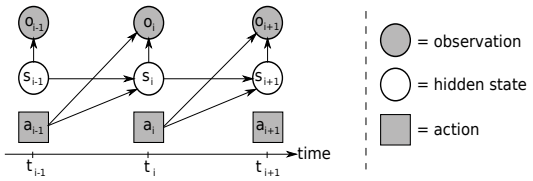

Figure 2. The POMDP model. Arrows indicate probabilistic influence.

The parameter $\mathscr{S}$ is a set containing the discrete states in which the environment can exist. In the TIS, the state of the environment is a representation of the key in which the musician is playing. There are 25 states, one for each musical key and an *inactive* state which indicates that the musician is not playing. The parameter $\mathscr{O}$ is a set containing the discrete observations that the agent can make. In the TIS, the observations are the possible outputs of the key-finding module. There are also 25 of these, one for each key and a *silent* observation to indicate that no notes have been played over the last two bars. Finally, the parameter $\mathscr{A}$ contains the actions available to the agent. In the TIS, the set of actions contains the possible instructions that can be sent to the improvisation-generating module. As described in Section 2, there are 25 of these, one for each key as well as the non-specific key instruction.

The parameter $\mathfrak{T}$ is the *transition probability function*. This describes the probabilistic dependance of the state of the environment at time t_{i+1} on the state at time t_i and the action taken at time t_i. That is, $\mathfrak{T}(s_{i+1}, s_i, a_i) = P(s_{i+1}|s_i, a_i)$, where $P(x)$ denotes the probability of x. The transition probabilities can be set by hand, or by some algorithm which learns the behaviour of the environment. In the TIS, they were set by hand such that it is most likely that the musician will remain in the same key rather than change key. From the inactive state, the transition to any other state is equally likely. There is also an increased likelihood that the musician will change to, or remain in the key played by the TIS.

The parameter $\mathfrak{Z}$ is the *observation probability function*. This describes the probabilistic dependance of the observation made at time t_i, on the state at time t_i and the action taken at time t_{i-1}. That is, $\mathfrak{Z}(o_i, s_i, a_{i-1}) = P(o_i|s_i, a_{i-1})$. For the TIS, the observation probabilities were simplified so that $\mathfrak{Z}(o_i, s_i, a_{i-1}) = P(o_i|s_i)$, i.e. the observations were made independent of the previous action. The probability

of making the silent observation when the musician is in the inactive state was set to unity. For the other observations, the vales of $P(o_i|s_i)$ were set by estimating the error rates of the key-finding module. This was done as follows. For each state s, corresponding to a key k_s, five thousand sequences of 16 equal-length notes were generated. Each sequence was generated by using the Krumhansl key-profile (see [5]) for k_s as a discrete probability distribution and sampling from it 16 times. The key-finding module was then used to estimate the key of each sequence. The probability $P(o|s)$ of making observation o, corresponding to key k_o, when the musician is in state s, was set equal to $N_{k_o}/5000$, where N_{k_o} was the number of times the key-finder gave k_o as its estimate. The observation probabilities obtained in this way for the keys C-major and C-minor are shown in Figure 3.

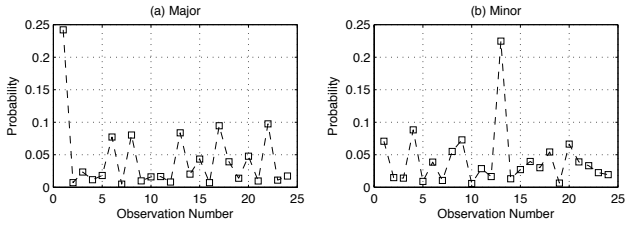

Figure 3. The estimated probabilities of each key being observed when the actual key is (a) C-major and (b) C-minor. Observations 1-12 are the 12 major keys, starting with C, and 13-24 are the 12 minor keys, starting with C.

The parameter $\mathfrak{R}$ is the *reward function*. This is a real-valued function of the current state and the chosen action, $\mathfrak{R}(s,a)$ and it describes the benefit of the agent performing action a when the environment is in state s. The agent uses the reward function to choose the optimal action to take (see below). In the TIS, the reward function was configured so that there is a positive reward, r^+, for playing in the same key as the musician, and a negative reward, r^-, for playing in a different key. A reward of zero was set for the non-specific action, independent of the state.

Usually, an agent acts so as to maximise the rewards obtained for actions taken over some time in the future. In many situations the rewards gained for future actions are considered less valuable than those gained for immediate actions. To allow for this, an additional parameter is included in the POMDP model. It is called the *discount*, γ ($0 \leq \gamma \leq 1$), and it describes the value of future rewards relative to immediate ones. A value of 1 means that future rewards are considered to be equal in value to immediate ones. A value of 0 means that only the immediate rewards have value. Non-zero discount models are used in situations where the agent is required to take into account the future states of the environment. The TIS is only required to play in the correct key at the current time, so a discount of zero is used.

Finally, a POMDP-based agent operates by maintaining a belief state, $b(s)$, which is a discrete probability distribution over the states in which the environment might exist. The parameter b_0 is the initial belief state. After each action is taken and a new observation is made, the belief state is updated (see [4] for details). To choose an action, the agent refers to a *policy* which is a mapping from belief state to action. The optimal policy is one which will maximise the rewards obtained. In general, finding the optimal policy is a computationally demanding task, albeit one which can be done offline (see, e.g. [10]). However, for models with $\gamma = 0$, such as that used in the TIS, the optimal action is simply the one which maximises the expected immediate reward, given by $\Sigma_{s \in \mathscr{S}} b(s).r(s,a)$.

4. SOFTWARE IMPLEMENTATION

To implement the key-choosing module, we developed a new object for Max called *pomdp.decider*. This object is used as follows. First, a POMDP model must be created and stored in a text file using the file format developed by Cassandra[3]. This file is then loaded into an instance of the pomdp.decider object in Max. If the model has a discount of zero, pomdp.decider can calculate the optimal action without a policy being explicitly set. Otherwise, a policy must be supplied by the user. There are a number of freely available POMDP solvers which can calculate optimal policies, including *zmdp* [10]. The pomdp.decider object has the ability to read policy files in the format output by zmdp. Once the model and policy (if it is required) have been loaded, pomdp.decider can be used. It takes integers representing observations as input and outputs integers indicating actions to take. The pomdp.decider object will be made freely available, along with more detailed usage instructions.

5. USING THE TONAL IMPROVISATION SYSTEM

The TIS is a toy system developed to experiment with the use of POMDPs in an IMS. Our aim was to investigate (i) how the reward function could be used to control the behaviour of the system, and (ii) how the transition and the observation probability functions would allow the system to take into account knowledge of the musician's behaviour and the accuracy of the key-finding module, respectively. This was done by informal experimentation with the system.

To investigate the effect of changing the reward function, the magnitude of r^- (see Section 3) was varied. When it was increased, the system tended to choose the non-specific key action unless the belief state had a peak close to unity for a particular key. When it was decreased, the system tended to choose the key with the highest probability, even if that probability was not close to unity. Additionally, the system

[3] http://www.cassandra.org/pomdp/code/pomdp-file-spec.shtml

could be made to tend towards a particular key by reducing the magnitude of r^- for that key.

In the TIS, the transition probability function describes the tendency of the musician to remain in the same key and the effect of this is to dampen the change in the belief state when a new observation is made. This was observed by varying p_s, which is the probability that the musician will remain in the same key. With p_s set close to unity, the belief state changed very little, and as p_s was reduced the damping decreased.

The effect of the observation probability function was studied by observing how the belief state changed for different sequences of observations. For example, if five C-Major observations are made and followed by an A-minor observation, the belief state changes very little. This is because observation probability function accounts for the tendency of the key-finding module to misidentify C-Major as A-minor (see Figure 3). However, if the five C-Major observations are followed by a C#-minor observation, the change in the belief state is greater, since the key-finding module is less likely to misidentify C-Major as C#-minor.

6. DISCUSSION AND CONCLUSION

A POMDP model can be used to control musical behaviour in many situations. For example, in systems which have musical style classifiers as their sensors such as those described in [2] and [3], a POMDP model would take classifier accuracy into account and it could use prior knowledge of a musician's tendencies and responses to inform its choices. More generally, there are numerous advantages to using a POMDP model in an IMS. The primary advantage is that there are structured machine learning methods for finding the optimal policy for selecting actions. In other words, instead of having to focus on developing heuristic rules that define a control policy, the designer can focus on the concrete aspects of the system which are the reward function, the transition probability function and the observation probability function. These functions provide a conceptually simple way for knowledge to be explicitly programmed into the system. For example the transition probability function models the behaviour of the musician and the observation probability function models the accuracy of the musical classifier. In addition, the musical goals of the designer can be conveniently expressed using the reward function. In this way, the task of programming an IMS to behave in a particular way is made conceptually more straightforward and easier to fine-tune and maintain than would be the case if an ad-hoc set of heuristics were used.

For most applications of POMDPs the reward function is known, however for many interactive music systems it is quite likely that the reward function will have to be learned. For the TIS example in this paper, it was easy to establish an appropriate reward function: positive for playing in the correct key and negative for playing in the incorrect key. However, if an IMS was required to act in some difficult-to-define manner, such as adhere to a particular musical style, we expect that a training procedure would be required in which the system would learn the reward function as well as the environmental responses. Further research is required to examine the implications of learning the reward function.

7. REFERENCES

[1] N. Collins, "Towards autonomous agents for live computer music: Realtime machine listening and interactive music systems," *Phd Thesis, University of Cambridge*, 2006.

[2] R. Dannenberg, B. Thom, and D. Watson, "A machine learning approach to musical style recognition," *Proc. International Computer Music Conference*, 1997.

[3] W. Hsu, "Two approaches for interaction management in timbre-aware improvisation systems," *Proc. International Computer Music Conference*, 2008.

[4] L. Kaelbling, M. Littman, and A. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[5] C. L. Krumhansl, "Cognitive foundations of musical pitch," *Oxford University Press*, 1990.

[6] K. Lee and M. Slaney, "Acoustic chord transcription and key extraction from audio using key-dependent hmms trained on synthesized audio," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 291 – 301, Feb 2008.

[7] G. Lewis, "Too many notes: Computers, complexity and culture in voyager," *Leonardo Music Journal*, vol. 10, pp. 33–39, Jan 2000.

[8] R. Rowe, "Interactive music systems," *MIT Press, Cambridge, MA*, 1993.

[9] S. J. Russell and P. Norvig, "Artificial intelligence: A modern approach," *Upper Saddle River, New Jersey: Prentice Hall*, 2003.

[10] T. Smith, "Probabilistic planning for robotic exploration," *PhD Thesis, Carnegie Mellon University*, 2007.

[11] D. Temperley, "Music and probability," *MIT Press, Cambridge, MA*, 2007.

[12] L. Yi and J. Goldsmith, "Decision-theoretic harmony: A first step," *International Journal of Approximate Reasoning*, pp. 1–12, Jul 2009.