The Use of Apprenticeship Learning Via Inverse Reinforcement Learning for Generating Melodies

Orry Messer
University of the Witwatersrand
orrymr@gmail.com

Pravesh Ranchod
University of the Witwatersrand
pravesh.ranchod@wits.ac.za

ABSTRACT

The research presented in this paper uses apprenticeship learning via inverse reinforcement learning to ascertain a reward function in a musical context. The learning agent then used this reward function to generate new melodies using reinforcement learning. Reinforcement learning is a type of unsupervised machine learning where rewards are used to guide an agent's learning. These rewards are usually manually specified. However, in the musical setting it is difficult to manually do so. Apprenticeship learning via inverse reinforcement learning can be used in these difficult cases to ascertain a reward function. In order to ascertain a reward function, the learning agent needs examples of expert behaviour. Melodies generated by the authors were used as expert behaviour in this research from which the learning agent discovered a reward function and subsequently used this reward function to generate new melodies. This paper is presented as a proof of concept; the results show that this approach can be used to generate new melodies although further work needs to be undertaken in order to build upon the rudimentary learning agent presented here.

1. INTRODUCTION

The task this research addresses is that of using machine learning techniques (specifically apprenticeship learning via inverse reinforcement learning) to generate melodies. The underlying concept used in this research is **reinforcement learning**. Using this approach to machine learning, an agent learns to perform a task by interacting with its environment. The task is conveyed to the agent using a reward function. The reward function maps states to rewards. If the agent reaches a goal state, it is given a positive reward. In the case of generating melodies, it is not clear what the reward signal should be. The approach taken in this research is to find a reward function, given expert behaviour. Learning then commences using the reward function. The expert behaviour in this context is a set of "expert" melodies (it is assumed that we have access to a set of expert melodies.) Learning in this manner is denoted **apprenticeship learning via inverse reinforcement learning** (AL via IRL) [1].

The use of AL via IRL stems from the idea that musical style is difficult to manually specify and that one gets the best sense of a musicians style by listening to the musician play. It is assumed in this research that a musician has a unique internal reward system which dictates how and when the musician will strike a note; this internal reward system governs how the musician plays. When the musician plays, he is therefore producing the output which maximizes his reward according to this reward system. The use of AL via IRL in this context is thus an attempt by a learning agent to learn a reward function which explains the expert's behaviour and thereafter to use this system to generate music.

Melody Agent (MA) was the learning agent created for the task of generating new melodies given a set expert trajectories. The results of the experiments performed using MA show that it was capable of generating new melodies, although (as expected) these melodies are similar to the expert trajectories.

The rest of this paper is broken up as follows: Section 2.1 discusses reinforcement learning; Section 2.2 discusses IRL, which is used to uncover a reward function given observed behaviour; Section 2.3 discusses AL via IRL, in which the uncovered reward function is used for learning; Section 3 then discusses other work in which reinforcement learning was used in a musical context. Section 4 describes the implementation of MA — its action space, state space as well as the expert melodies used. Section 5 describes the results of the two experiments performed using MA. In Sections 6, 7 and 8 a discussion on the results, conclusions and future work are presented, respectively.

2. BACKGROUND

2.1 Reinforcement Learning

Reinforcement learning is a form of unsupervised machine learning wherein the learning agent interacts with the environment in order to achieve a goal [2]. The learning agent interacts with its environment by taking an **action** which changes the **state** of its environment. A state is a configuration of the environment and an action is how an agent affects state. Taking actions which do not only look to maximize immediate reward, but rather to maximize future rewards is an important feature of reinforcement learning. The agent looks to maximize future rewards by directing itself to states from which more reward can be gained; that is, it tries to direct itself to states with high **value**. Value can be thought of as "how good" it is to be in a particular state, with respect to the expected future reward. A state only one step away from a goal state would have a higher value than a state two steps away from that same goal state. A **value function** maps states to values. A related concept, and one that is more pertinent to this research, is that of the **action value function**. This function describes the value of taking an action within a state. This is discussed further in section 2.1.2.

How an agent behaves - that is, which actions it takes in which states - is dictated by a **policy**. A policy is a mapping from states to probabilities of selecting actions. A policy may be deterministic in which case the policy is a mapping from each state to a corresponding action. A policy is denoted by the letter π.

2.1.1 Return

The learning agent attempts to ascertain the optimal policy - this is the policy which maximizes the expected cumulative discounted reward from a given state. The cumulative discounted reward is known as the **return**. This is shown in equation 1.

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \qquad 0 \leq \gamma \leq 1 \tag{1}$$

In the above equation, γ is the discount rate which is between 0 and 1. It is used in order to ensure that the sum in 1 has a finite value. The closer γ is to 1 the more far sighted the agent becomes, weighing future rewards more heavily. If γ is 0 then the agent concerns itself only with immediate reward, not taking into account any future rewards received.

Note here that this is the actual return received by the learning agent. If the environment's dynamics were known, this return could be calculated a priori. What the learning agent attempts to uncover is a policy which will direct the agent in such a way that R_t is maximized. Of course, this problem is non-trivial as the dynamics of the environment can be (and usually are) complex, unknown and laden with uncertainty.

2.1.2 Action Value Function

The action value function for a policy π is defined as follows:

$$Q^{\pi}(s,a) = E_{\pi}\{R_t | s_t = s, a_t = a\} \tag{2}$$
$$= E_{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s, a_t = a \right\} \tag{3}$$

where $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$ is the return (that is, the cumulative discounted reward), following time t. The action-value function associates to each state the value of each action from that state, under a given policy. The next section presents Sarsa - an algorithm which estimates the action value function in order to find an optimal policy.

2.1.3 Sarsa

Figure 1 presents Sarsa which is used to find a policy which will elicit good behaviour from a learning agent, where good behaviour is measured relative to a reward function.

```
1:  Initialize $Q(s,a)$ arbitrarily
2:  **loop** (for each episode):
3:      Initialize s
4:      Choose $a$ from $s$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
5:      **repeat** (for each step of episode):
6:          Take action $a$, observe $r$, $s'$
7:          Choose action $a'$ from $s'$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
8:          $Q(s,a) \leftarrow Q(s,a) + \alpha\big[r + \gamma Q(s',a') - Q(s,a)\big]$
9:          $s \leftarrow s'$; $a \leftarrow a'$;
10:     **until** s is terminal
11: **end loop**
```

Figure 1. The Sarsa Control Algorithm

This reward function is external to the learning agent; it is part of the environment but the implementer of the algorithm must still specify this reward function. It is the way in which the goal of the task is conveyed to the agent. There are times, however, when it is difficult or impossible to manually specify the reward function [1]. The next section discusses inverse reinforcement learning — the task of finding a reward function.

2.2 Inverse Reinforcement Learning

Inverse reinforcement learning is the problem of uncovering a reward function. It has been characterized in [3] as follows: **Given** 1) measurements of an agent's behaviour over time, in a variety of circumstances, 2) if needed, measurements of the sensory inputs to the agent; 3) if available, a model of the environment — **Determine** the reward function being optimized.

Two motivations for inverse reinforcement learning were described in [4]; the first being its use in uncovering computational models for animal and human learning. This has also been applied to human economic behaviour. This first case, used when examining the behaviour of agents, is denoted *reward learning*. The second motivation is that of *apprenticeship learning*. As in the reward learning case, the reward function is ascertained. The extension in the apprenticeship learning case is that once the reward function is found, it is used to direct the learning of a learning agent. Thus it can be said that the uncovered reward function describes the behaviour of an expert (although the algorithm need not retrieve the exact reward function used by the expert.) The following section describes apprenticeship learning in more detail.

2.3 Apprenticeship Learning via Inverse Reinforcement Learning

In a traditional reinforcement learning task, the reward function is specified manually and a (near) optimal policy is found by the learning agent in order to maximize a numerical reward signal. In the AL via IRL case, an expert is thought to have some reward function which she is trying to maximize. The algorithm used to perform the task of AL via IRL (presented in Section 2.3.3, originally published in [1]) does not necessarily recover the expert's reward function. However, it will return a reward function which can explain the expert behaviour and induce behaviour similar to that of the expert's, assuming adequate learning is done using the uncovered reward function. In order for the algorithm to work, it needs a set of *expert trajectories*. An expert trajectory is a set of states; it is a set of states which the expert observed while navigating its environment. It is assumed that the expert has behaved a desirable manner and thus the expert trajectory is a "good" trajectory through an environment. Expert trajectories are discussed further in the next section in the context of reward functions.

2.3.1 Reward Function

The reward function can be represented as the scalar result of the dot product of a set of weights, w, and the feature vector of a state, $\phi(s)$ as shown in equation 4. In implementing apprenticeship learning, the reward function to be recovered is represented as this equation.

$$R(s) = w \cdot \phi(s) \tag{4}$$

The feature vector, ϕ, contains the state variables which provide information about the environment's current configuration. If the agent is in state s_k, then the features of the state will be $\phi(s_k)$. It is the weight vector, w, which then determine the reward an agent receives in a particular state. A different weight vector will thus yield a different reward function. Thus, the apprenticeship learning algorithm must return an appropriate weight vector — one which characterizes a reward function which led the expert to behave in the observed way.

As previously discussed, an expert trajectory is a walk through the state space performed by the expert. The expert is guided through the state space by its *expert reward function*. It is assumed that the expert's reward function is made up of an optimal weight vector, w^* such that:

$$R^*(s) = w^* \cdot \phi(s) \tag{5}$$

where $R^*(s)$ is the expert's reward function.

The goal of apprenticeship learning algorithm is to retrieve a weight vector for characterizing a reward function which can explain the expert trajectories. In order to do so, the algorithm makes use of the expert's **feature expectations** which are discussed in the following section.

2.3.2 Feature Expectations

The value of a policy π may be written as follows:

$$E_{s_0 \sim D}[V^\pi(s_0)] = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \middle| \pi\right] \tag{6}$$

$$= E\left[\sum_{t=0}^{\infty} \gamma^t w \cdot \phi(s_t) \middle| \pi\right] \tag{7}$$

$$= w \cdot E\left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t) \middle| \pi\right] \tag{8}$$

Where s_0 is drawn from D, the set of starting states. The step from 6 to 7 follows from equation 4. Equation 8 may be rewritten as:

$$E_{s_0 \sim D}[V^\pi(s_0)] = w \cdot \mu(\pi) \tag{9}$$

where

$$\mu(\pi) = E\left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t) \middle| \pi\right] \tag{10}$$

$\mu(\pi)$ is the vector of **feature expectations** of a particular policy. It is defined as the expected discounted accumulated feature value vector [1]. It represents a discounted sum of the feature vectors which are expected to be seen when following a particular policy. In order to ascertain the feature expectations, given a policy, Monte Carlo methods may be used to sample trajectories, or if the dynamics of the environment are known and it is tractable, they may be computed [1].

The apprenticeship learning algorithm relies on knowing the *expert*'s feature expectations; that is knowing μ_E. In practice an estimate for the expert's feature expectations, $\hat{\mu}_E$ is found empirically; this can be done since we assume access to expert trajectories. An expert trajectory is the expert's path through the state space: $\{s_0, s_1, \ldots\}$. Suppose there are m trajectories through the state space, then we have: $\{s_0, s_1, \ldots\}_{i=1}^{m}$. The empirical estimate for μ_E is given by equation 11.

$$\hat{\mu}_E = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{\infty} \gamma^t \phi(s_t^{(i)}) \tag{11}$$

2.3.3 Apprenticeship Learning Algorithm

Once the expert feature expectations have been estimated, the apprenticeship learning algorithm attempts to find a policy, $\tilde{\pi}$, such that $\tilde{\pi}$ induces feature expectations close to $\hat{\mu}_E$ [1].

The apprenticeship learning algorithm for doing so works as follows:

```
1: Randomly pick some policy $\pi^{(0)}$, compute (or approximate via Monte Carlo) $\mu^{(0)} = \mu(\pi^{(0)})$, and set $i = 1$.
2: Set $w^{(1)} = \mu_E - \mu^{(0)}$ and $\bar{\mu}^{(0)} = \mu^{(0)}$
3: Set $t^{(1)} = ||w^{(1)}||_2$
4: if $t^{(1)} \leq \epsilon$ then
5:     terminate
6: else
7:     loop (while $t^{(i)} > \epsilon$):
 8:         Using the RL algorithm, compute the optimal policy $\pi^{(i)}$ for the MDP using $R = (w^{(i)})^T\phi$
 9:         Compute $\mu^{(i)} = \mu(\pi^{(i)})$
10:         Set $i = i + 1$
11:         Set $a = \mu^{(i-1)} - \bar{\mu}^{(i-2)}$
12:         Set $b = \mu_E - \bar{\mu}^{(i-2)}$
13:         Set $\bar{\mu}^{(i-1)} = \bar{\mu}^{(i-2)} + \frac{a^T b}{a^T a} a$
14:         Set $w^{(i)} = \mu_E - \bar{\mu}^{(i-1)}$
15:         Set $t^{(i)} = ||w^{(i)}||_2$
16:     end loop
17: end if
```

[1] What this algorithm tries to accomplish is to find a policy such that the agent's performance under that policy is close to that of the expert's under the unknown reward function $R^*(s) = w^* \cdot \phi(s)$. Such a policy $\tilde{\pi}$ will have $||\mu(\tilde{\pi}) - \mu_E||_2 \leq \epsilon$, where ϵ is an arbitrarily small positive number. This means that the feature expectations which are induced by such a policy will be arbitrarily close to the feature expectations of the expert.

3. RELATED RESEARCH - REINFORCEMENT LEARNING AND MUSIC

Reinforcement learning depends on having a clearly defined reward function which the agent then uses to guide its learning. In a musical setting, it is not at all straightforward what the reward signal should be. In the paper *Reinforcement Learning for Live Musical Agents*, [5] presents three ideas for musically meaningful reward signals.

The first idea is to match internal goals by the imposition of a rule set. This has been explored by [6, Franklin et al.] who based the reinforcement signal on eight handwritten rules for jazz improvisation. They used actor-critic reinforcement learning using a non-linear recurrent neural network for note generation. The second idea [5, Collins] presents is to base the reward signal on the appreciation of fellow participants (other musicians with whom the learning agent is performing) and audience members. He suggests the use of tracking facial expressions and using physiological indicators such as monitoring galvanic skin response to gauge a listeners engagement, although these ambitious approaches have not yet been explored. A third reward signal he proposes is *memetic success*, that is, the taking up of the agent's musical ideas by others. This approach was explored by [5, Collins] using his music framework *Improvagent*, where the learning agent improvised with fellow musicians basing its reward on how much it influenced the position in the state space, given its choice of action; the effect of the action it took is measured by whether or not it influenced the current status quo of the musical piece (as played by the fellow musicians).

In another approach, the OMax system [7] used reinforcement learning to weight links in a Factor Oracle (FO). A FO is a finite state automaton constructed incrementally in linear time and space. A musical sequence is used to build up the FO. Each symbol in the sequence corresponds to a state in the FO and reinforcement signals are used to weight the links between these states. This can be used for live musical interaction.

Most relevant to this research is the approach taken by [8], in which apprenticeship learning via inverse reinforcement learning was also used. The expert trajectories given to the learning algorithm were Bach's Chorales; the system then managed to create original melodies whose overall shape was characteristic of Bach's work. The research presented in this document differs from that presented in [8] in three ways.

Firstly, the music given to the system as expert trajectories in this research was generated by the author, as opposed to being given Bach's Chorales as was done in the work presented in [8].

Secondly, the state signal differed. A full discussion of the state signal is deferred until Section 4.3, but briefly, the state signal encoded the last eight actions taken, where an action was a choice of note. In the [8] case, the state signal was a tuple consisting of the position within the musical piece, the current pitch of the melody, the difference between the current pitch and the pitch of the previous state, the current chord type, the difference between the current chord type and the root of the previous chord and finally the status of the melody: whether it is resting, continuing to sound a previous note or starting a new note.

The third difference is in the action space. The actions available to MA where which note to play next. A more comprehensive discussion is deferred to section 4.1. The actions in the research by [8] denoted whether there was a change in the current portion that the musical piece was in, whether there was a change in the pitch of the melody, whether there was a change in the root of the chord being played, the resulting chord type from taking the action and finally, the status of the resulting musical state: whether this action is going to rest, hold or state a new note.

4. RESEARCH METHODOLOGY

4.1 Actions

The actions available to MA correspond to a range of notes which the agent could play at each time step in the musical piece. Each note was given a corresponding action number; action 0 was a rest, action 1 was a D, action 2 was $D\sharp3/E\flat3$ and so on. In total, a 2 octave range of notes was used as the action space for the learning agent. Thus, there were 25 actions available to MA. This is as a result of a range of 2 octaves, each containing 12 notes, as well as the rest action, in which no note is played.

4.2 Expert Trajectories

The melodies presented to MA as expert trajectories were 8 measures long with each measure containing 8 beats, yielding a total of 64 possible positions in which an action may be taken (here taking an action refers to playing a note). One of the expert trajectories is shown in figure 2. This figure shows the expert trajectory as a series of action numbers. Each row in the matrix represents a measure and each column represents a beat within that measure. Since each note is held for one eighth of the measure, each action corresponds to playing a note for an eighth note. Figure 2 shows this same melody as a series of notes and figure 4 shows the melody as sheet music.

12	13	15	16	18	0	18	0
12	13	15	16	18	0	18	0
18	16	15	13	12	0	12	0
18	16	15	13	12	0	12	0
18	20	21	24	25	0	25	0
25	24	21	20	18	0	18	0
18	20	21	24	25	0	25	0
25	24	21	20	18	0	18	0

Figure 2. An expert melody as series of action numbers. The rows correspond to the measures and the columns to the beats within the measures.

$C\sharp 4$	$D4$	$E4$	$F4$	$G4$	$-$	$G4$	$-$
$C\sharp 4$	$D4$	$E4$	$F4$	$G4$	$-$	$G4$	$-$
$G4$	$F4$	$E4$	$D4$	$C\sharp 4$	$-$	$C\sharp 4$	$-$
$G4$	$F4$	$E4$	$D4$	$C\sharp 4$	$-$	$C\sharp 4$	$-$
$G4$	$A4$	$B\flat 4$	$C\sharp 5$	$D5$	$-$	$D5$	$-$
$D5$	$C\sharp 5$	$B\flat 4$	$A4$	$G4$	$-$	$G4$	$-$
$G4$	$A4$	$B\flat 4$	$C\sharp 5$	$D5$	$-$	$D5$	$-$
$D5$	$C\sharp 5$	$B\flat 4$	$A4$	$G4$	$-$	$G4$	$-$

Figure 3. This figure shows the same expert melody in figure 2 as a series of notes.

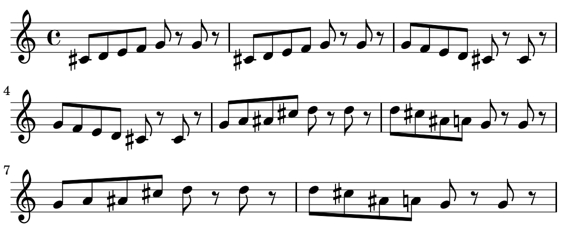

Figure 4. This figure shows the sheet music corresponding to the expert trajectory shown in figures 2 and 3

The Melody Agent (MA) is a melody generating implementation of the apprenticeship learning algorithm. It received a state signal composed of the last 8 actions it took. An action in this context refers to a particular choice of note.

4.3 States

The state signal which MA received was comprised of the last eight actions it took. Thus, the following 8-tuple was received by the agent as a state signal:
$(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7)$. Each x_i refers to a previous action taken, where x_0 was the most recent action taken and x_7 the least recent. The state space of MA was 25^8. This is as a result of the 25 actions available to MA.

These eight state variables were used to make up the feature vector, ϕ. Thus, the feature vector encoded the previous eight actions taken by the learning agent. The starting state of the agent was one in which no actions had been recorded in any of the last eight possible positions - each element in the state signal had a value of 0. That is, the initial state signal was $(0, 0, 0, 0, 0, 0, 0, 0)$.

5. RESULTS

The parameters used for MA were as follows:

- $\gamma = 1$
- $\gamma_2 = 0.8$
- $\epsilon = 0.15$
- $\epsilon_2 = 0.05$
- $\alpha = 0.08$
- Sarsa was given 100000 episodes [1]

γ refers to the discount rate for calculating the feature expectations. The discount rate used in the Sarsa update rule is γ_2. ϵ was the exploration rate used by Sarsa and α was its learning rate. ϵ_2 was the stop condition used for the apprenticeship learning algorithm, although this was not used - for both experiments the program was stopped manually once several policies had been produced.

5.1 Experiment One

The purpose of this experiment was to test whether MA could generate new melodies given a set of expert melodies. The set of expert melodies used as expert trajectories can be heard at `http://aiml.cs.wits.ac.za/orry/audio.html`. All of these expert trajectories are in the key of D harmonic minor. Figures 5 - 8 illustrate how a new melody is generated. Figure 5 is the new generated melody. The red, blue and green highlighted portions of the melody are all phrases which can be observed in one of the expert melodies. The sheet music corresponding to this generated melody is shown in figure 9. Each coloured phrase corresponds to a phrase highlighted in the same colour in figures 6 - 8 (figures 6 - 8 show expert melodies). MA has learned that the states which have been observed in the expert trajectories are "good" states and thus it learns to navigate its was to these states.

8	13	15	8	13	15	8	8
8	13	16	8	13	16	13	4
4	8	13	4	8	13	8	1
1	8	13	16	20	24	25	0
13	16	20	13	0	0	0	0
0	0	1	3	4	6	3	4
6	8	4	6	8	9	6	8
0	0	0	0	0	0	0	8

Figure 5. This generated melody can be heard at `http://aiml.cs.wits.ac.za/orry/Agents/MA/Exp_1/1.mp3`

[1] An episode is one full iteration of the Sarsa algorithm.

8	13	15	8	13	15	8	8
8	13	16	8	13	16	8	8
8	13	15	8	13	15	9	8
8	13	18	8	13	18	9	9
8	13	15	8	13	15	8	8
8	13	16	8	13	16	8	8
8	13	15	8	13	15	9	8
8	13	18	8	13	18	9	9

Figure 6. This expert melody can be heard at `http://aiml.cs.wits.ac.za/orry/Expert_Trajectories/Melody/t5.mp3`

13	16	20	13	16	20	16	8
8	13	16	8	13	16	13	4
4	8	13	4	8	13	8	1
1	8	13	16	20	24	25	0
13	16	20	13	16	20	16	8
8	13	16	8	13	16	13	4
4	8	13	4	8	13	8	1
1	8	13	16	20	24	25	0

Figure 7. This expert melody can be heard at `http://aiml.cs.wits.ac.za/orry/Expert_Trajectories/Melody/t6.mp3`

1	3	4	6	3	4	6	8
4	6	8	9	6	8	9	12
8	9	12	13	9	12	13	15
12	13	15	16	13	15	16	18
15	16	18	20	16	18	20	21
18	20	21	24	20	21	24	25
25	20	24	21	20	21	24	25
0	0	0	0	0	0	0	0

Figure 8. This expert melody can be heard at `http://aiml.cs.wits.ac.za/orry/Expert_Trajectories/Melody/t7.mp3`

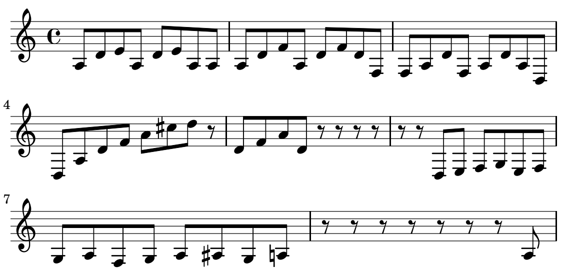

Figure 9. This figure shows the sheet music corresponding to the generated melody shown in figure 5

5.2 Experiment Two

In this experiment, the learning agent used ten D harmonic minor expert trajectories along with an additional ten melodies in the key of D major. These trajectories can be heard at `http://aiml.cs.wits.ac.za/orry/audio.html`. Two new melodies were generated in this experiment. Both of these melodies contained phrases from the D-harmonic minor as well as the D-major expert trajectories. These melodies can be heard at `http://aiml.cs.wits.ac.za/orry/Agents/MA/Exp_2/1.mp3` (sheet music shown in figure 10) and `http://aiml.cs.wits.ac.za/orry/Agents/MA/Exp_2/2.mp3` (sheet music shown in figure 11).

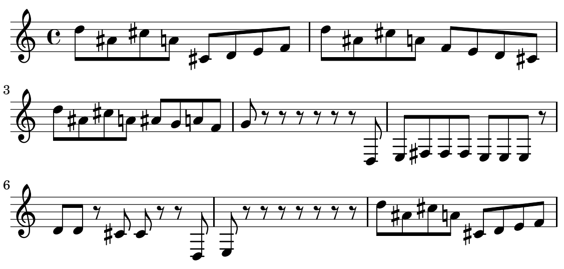

Figure 10. A generated melody.

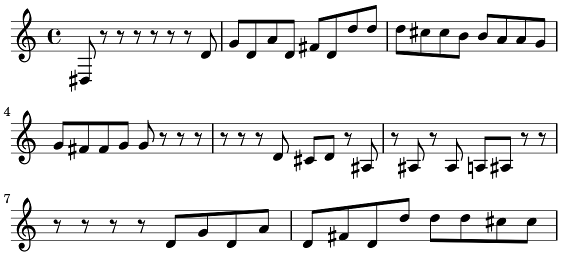

Figure 11. Another generated melody.

6. DISCUSSION

The results shown in the previous section came from one run of the apprenticeship learning algorithm. These results show that the apprenticeship learning algorithm is capable of producing original melodies which are similar to the expert melodies. MA shows signs of learning but it appears that the start state of $(x, 0, 0, 0, 0, 0, 0, 0)$ (where x is an action) has influenced learning. Since the state of the learning agent was the last eight notes played, the first state of all the expert trajectories was $(x, 0, 0, 0, 0, 0, 0, 0)$, where $0 \leq x \leq 24$ is an action. This is because no notes have been played prior to the first one, and as a result all actions prior to the first one are encoded as zeroes. That is, they are the action in which no note has been played. This has affected learning; as can be seen from the results, the pattern $0000000x$, where $0 \leq x \leq 24$ is an action is prevalent in the generated melodies. This is likely due to the fact that the algorithm considers this a good state to be in, as it was observed in all of the expert trajectories.

Although the corpus of expert trajectories was limited to D harmonic minor in the first experiment and D major and D harmonic minor in the second, it is not necessary for the expert trajectories to be in these keys. The choice of using only D major in the first experiment was to test whether the output would be predominantly in this key, which it was. In the second experiment, it was hoped that the resulting melodies would be in either D major or D harmonic minor. The result was that the melodies exhibited phrases from both the D major and the D harmonic minor corpora. This is likely due to the state signal not conveying enough information about the melodies; the learning agent has no way of knowing that it is "good" to stay in a particular key.

These experiments represent a first attempt at using AL via IRL with the conceptually simple action and state spaces discussed in Sections 4.1 and 4.3 respectively. Although the state space will likely need to change in order for the learning agent to have some indication of key, it is hoped that the state space need not be overly complicated.

7. CONCLUSIONS

This paper provides the initial results experiments done using MA. The results show that given a set of expert trajectories, MA can create new melodies which are stylistically similar to the expert trajectories. These results show that implementing apprenticeship learning via inverse reinforcement learning in this way can lead to a learning agent which can generate new melodies which are different to the expert melodies whilst still maintaining those melodies' characteristics. The results show that the algorithm can return generated melodies which clearly resemble the expert melodies, but are put together in unexpected ways. This work is presented as a proof of concept, rather than a final implementation of this approach.

8. FUTURE WORK

Apprenticeship learning via reinforcement learning shows potential as a way to algorithmically generate music. The results presented in this paper show that it is possible to use this algorithm to create new melodies based on expert melodies. The approach taken in this research can be improved upon in many ways. The most obvious way to extend this research is to present the learning agent with more expert trajectories. As more trajectories are added, the learning agent will have a greater base from which to extract the rewards which guided the creation of those trajectories. Another way the research can be extended is by providing a deeper note resolution; currently, the learning agent could only play eighth notes; this could be extended so that the agent could play sixteenth notes, allowing for more variety in the type of music it can handle. Further, the agents could be extended to play triplets, quintuplets and even septuplets.

Another interesting approach is to create multiple music agents and to allow the learning agents to learn in a way that promotes collaborative music generation; for example, if its seen that whenever a bass drum is played, a certain note is played, provide a mechanism for the learning agent to take that into account. There are many approaches one could take to implement a learning agent which makes use of reinforcement learning for music generation. This research presents one such approach which has been shown to generate new melodies which are unpredictable but coherent.

9. ACKNOWLEDGEMENTS

The authors wish to acknowledge financial assistance from the National Research Foundation of South Africa's Thuthuka Programme.

10. REFERENCES

[1] P. Abbeel and Y. Ng, A, "Apprenticeship learning via inverse reinforcement learning," Computer Science Department, Stanford University, Stanford, CA 94305, USA, Tech. Rep., 2004.

[2] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998. [Online]. Available: http://www.cs.ualberta.ca/%7Esutton/book/ebook/the-book.html

[3] S. Russell, "Learning agents for uncertain environments (extended abstract," in *In Proceedings of the eleventh annual conference on Computational learning theory (COLT98*, 1998.

[4] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *in Proc. 17th International Conf. on Machine Learning*. Morgan Kaufmann, 2000, pp. 663–670.

[5] N. Collins, "Reinforcement learning for live musical agents," in *Proceedings of ICMC2008, International Computer Music Conference, Belfast*. Citeseer, 2008.

[6] J. A. Franklin and V. U. Manfredi, "Computational intelligence and applications," A. Abraham, B. Nath, M. Sambandham, and P. Saratchandran, Eds. Atlanta, GA, USA: Dynamic Publishers, Inc., 2002, ch. Nonlinear Credit Assignment for Musical Sequences, pp. 245–250. [Online]. Available: http://dl.acm.org/citation.cfm?id=989710.775004

[7] G. Assayag, G. Bloch, M. Chemillier, A. Cont, and S. Dubnov, "Omax brothers: a dynamic topology of agents for improvization learning," in *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, ser. AMCMM '06. New York, NY, USA: ACM, 2006, pp. 125–132. [Online]. Available: http://doi.acm.org/10.1145/1178723.1178742

[8] J. Acevedo, S. Gliske, and M. Jayap, "Musical style replication using apprenticeship learning," 2007. [Online]. Available: http://web.eecs.umich.edu/~cscott/past_courses/eecs545f07/projects/AcevedoGliskeJayapandian.pdf