Directed Transitional Composition for Gaming and Adaptive Music Using Q-Learning

Jason Cullimore
Interdisciplinary Studies
University of Regina
jason@jasoncullimore.com

Howard Hamilton
Computer Science
University of Regina
hamilton@cs.uregina.ca

David Gerhard
Computer Science
University of Regina
gerhard@cs.uregina.ca

ABSTRACT

One challenge relating to the creation of adaptive music involves generating transitions between musical ideas. This paper proposes a solution to this problem based on a modification of the Q-Learning framework described by Reese, Yampolskiy and Elmaghraby. The proposed solution represents chords as states in a domain and generates a transition between any two major or minor chords by finding a pathway through the domain in a manner based on a Q-Learning framework. To ensure that the transitional chords conform to the tonalities defined by the start and goal chords, only chords that contain notes that are found in combined pentatonic scales built from the start and goal chords are included within the domain. This restriction increases the speed of pathfinding and improves the conformation of the transitions to desirable tonal spaces (in particular the keys most closely related to the start and goal chords). This framework represents an improvement over previous music generation systems in that it supports transitions from any point in a musical cue to any point in another, and these transitions can be rendered in real time. A general method for implementing this solution in a video game is also discussed.

1. INTRODUCTION

Reese, Yampolskiy and Elmaghraby [1] present a framework whereby a Q-Learning model can be applied within a process that generates sequences of musical chords. We propose to modify their framework to generate musical sequences that express major and minor tonalities more strongly, with the goal being the creation of transitional chord sequences linking any two major or minor chords. We will also discuss ways to increase the computational efficiency of sequence generation as well as the aesthetic quality of generated chord sequences through modifying Reese *et al.*'s model to restrict the range of chord types available to the generative process.

Development of such a system would be of great use to both composers and the video game developers who employ them. As video games become more complex, sometimes reaching play times of over 100 hours, they require more varied musical scores to keep the experience fresh and engaging for the player. Meeting this challenge can require a great deal of composing, both of musical cues and of transitional material to link these cues. For example, the component of the interactive score to *Grand Theft Auto V* produced by Edgar Freose [1] featured over 67 hours of music, including cues, transitions and alternate audio mixdowns [2]. This estimate excludes the total length of the commercial pop songs heard on the in-game radio.

Because of the large number of cues that are incorporated into a modern video game, flexible methods of smoothly transitioning between highly differing cues have become a significant area of technical development in the video game industry. There have been many solutions proposed for this problem. One early example is found in LucasArts' iMUSE interactive music system [3]. With iMUSE, the composer "... must map out and anticipate all possible interactions between sequences at any marker point, and compose compatible transitions ... an incredibly time consuming process." Marker points are places in a cue where the composer dictates that a change in the music may occur, such as a transition to another theme, an end to the music, or a continuation of previous musical material. The larger the game (and associated soundtrack), the greater the number of possible transitions and the more complex the task facing the composer, who is responsible for composing both the cues and transitions, while at the same time implementing an event-based decision tree that reflects all possible sequences in which the musical score may unfold.

Another approach was used in the game *Anarchy Online*, which featured a series of about 750 musical fragments called "sequences"[3]. These sequences could be layered and sequentially arranged into a continuous musical soundtrack. Groups of sequences expressing particular moods were associated with specific locations in the game world, giving each environment its own musical character. While this approach allows a score to change dynamically in a manner that can sustain many hours of playtime, the score is still limited in that it cannot autonomously generate new musical material that falls outside of the structures present within the precomposed sequences. Also, a composer using such a system is limited with regards to the size and scope of the themes that they create; musical themes that require development over many measures of music may be less compatible with a compositional approach that em-

[1] GTA5 is a video game that incorporates racing and role-playing elements released by Rockstar Games in 2013

phasizes short, interconnecting musical ideas.

The system we propose in this paper avoids such precomposed elements within transitional material. Instead, we adapt Reese *et al.*'s Q-Learning framework to generate transitions between cues automatically and at any point within the cues. The system also generates transitions that inhabit the same tonal space as the chords which form the start and end point of the transition. Such a system would be of great use to a composer of nonlinear, adaptive music, whether in video game scoring applications or the creation of more aesthetically focussed artworks, since it allows for the efficient and interactive generation of transitional material. Dynamically generated transitional music that is capable of reacting to changing elements within a game state (*e.g.* player actions, shifts in environmental attributes, and changes in the desired emotional state for the player at a given point in the gameplay) can be guided both by the composer during the game's development and by the game engine dynamically at run time. As a result, dynamic musical transitions can increase the flexibility with which the game engine can instantiate musical cues, since, theoretically, such a system would be able to transition from any cue (expressing one emotion) to any other cue (potentially evoking a highly contrasting emotion). The composer could then focus on writing expressive cues, knowing that these cues can be smoothly transitioned between in a manner that allows the music to closely follow the action in the game.

The remainder of this paper is organized as follows. First, A machine-learning algorithm known as *Q-Learning* will be presented. Next, a specific application of Q-Learning to chord generation will be discussed, and finally, a set of improvements and modifications will be proposed to the existing Q-Learning chord sequence generation method designed to allow it to operate in real time and generate appropriate, interesting, non-repeating transitional sequences starting at any chord in the original musical sequence.

2. Q-LEARNING

The Q-Learning model was first developed by Watkins [4] and refined by Watkins and Dayan [5]. Poole and Mackworth [6] present a useful description of the model.

In this model there is a *domain* which can be configured with different attributes at different times. Each configuration of the domain constitutes a *state*, which is a member of the total set of states that the domain can embody. At any given time, an *agent* is located at one of these states, and may traverse from one state to another within the domain by executing an *action*. Allowable actions for any given state are defined by a set of rules that are devised by the programmer. Figure 1 shows an example [2] of a simple six-state domain. The agent begins in state S0 (the start state) and is tasked with finding the optimal route from S0 to S5 (the goal state). The allowable actions from each state have been defined by the designer and are shown by the arrows in the figure.

[2] Figure 1 and the associated example are adapted from `www.acm.uiuc.edu/sigart/docs/QLearning.pdf`

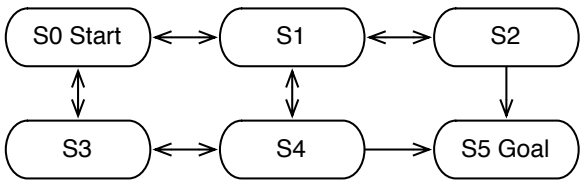

Figure 1. A hypothetical domain consisting of six states

Each action that the agent can take from a particular state can be represented as a pairing of a state and an action. It is possible to represent all pairs of state and action on a two dimensional matrix with the rows representing the origin states and the columns representing actions taken from an origin state that each lead to a specific, unique destination state. There are two such matrices used in the Q-Learning process:

- The *reward matrix* (R) contains fixed values that indicate the desirability of an action taken from a given state;
- The Q *matrix* records the cumulative reward values generated as the agent traverses the domain. These values are updated every time the agent executes a particular action from a particular state.

Table 1 shows a reward matrix and Table 2 shows a Q matrix for the domain in Fig. 1. Note that each state in both matrices is associated with several possible actions (based on a rule set defined by the programmer). In the case of the reward matrix, the empty cells indicate that the corresponding action is not permissible (*i.e.* there is no arrow that connects the two relevant states). All permissible actions have a reward value of zero, except the two transitions to the goal state, which each earn a reward of 100. In the Q matrix, all values may initially be zero, indicating that no learning has yet occurred. These values will become nonzero as the agent repeatedly moves through the domain and is rewarded (note that the initial Q matrix could also be populated with random values). As in the case of the reward matrix, non-allowed actions are excluded from the Q matrix in Table 2.

R	A0	A1	A2	A3	A4	A5
S0		0		0		
S1	0		0		0	
S2		0				100
S3	0				0	
S4		0		0		100
S5			0		0	

Table 1. A reward matrix for the domain in Fig. 1.

In the Q-Learning model, the agent begins in the start state (in this case state S0) and traverses the domain by executing allowable actions for whatever state the agent currently occupies. Each action within the domain is an event which results in an update to the value of $Q(s, a)$, based upon the current reward value listed in the reward

Q	A0	A1	A2	A3	A4	A5
S0		0		0		
S1	0		0		0	
S2		0				0
S3	0				0	
S4		0		0		0
S5			0		0	

Table 2. An initial Q matrix for the domain in Fig. 1

table, plus an additional value proportional to the Q matrix value of the most desirable future action available from the destination state. These values serve to differentially reward pathways based upon the desirability of particular state/action combinations. When an agent executes an action, the corresponding Q matrix value is updated. Higher values reinforce use of an action and lower values reduce its desirability.

In our example, the result of these repeated updates to the values in the Q matrix is that the rewards associated with actions that lead into the goal state are propagated back through the Q matrix. As such, state/action pairs that lead into the goal state also develop higher Q values, and state/action pairs that lead to states with higher Q values also themselves develop increased Q values. Ultimately, a path of relatively high Q values forms between the start state and the goal state. This path can be iteratively reinforced until the algorithm approaches convergence.

Entries in the Q matrix are updated using the following process: Given an action a from a state s with a reward $R(a, s)$, causing a change to state s' with available actions a', the new value for the $Q(s, a)$ is calculated according to the formula [1]:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a)+\alpha\left(R(s,a)+\gamma\max_{a'}Q(s',a')\right) \quad (1)$$

where $(0 < \alpha \leq 1)$ represents the learning rate (the degree to which new information overrides the current $Q(s, a)$ value), and $(0 \leq \gamma < 1)$ is the the discount factor, (the degree to which the agent values future rewards).

For each cycle of the Q-Learning algorithm, the agent:

1. Identifies the actions available in the current state;
2. Selects one at random to execute;
3. Calculates the total reward for executing this action (as in Eq. 1); and
4. Updates the $Q(s, a)$ value for the original state and action taken.

The agent then executes this algorithm again with the destination state as the new starting point, continuing in this manner until the agent reaches the terminal state, completing a learning *episode*. Further episodes are executed and the Q matrix is iteratively updated until the matrix converges upon an optimal solution for traversing the states in the domain, known as an *optimal policy*. As this happens, each state/action pair will then approach an optimal Q value $Q^*(s, a)$. Full convergence theoretically requires an infinite number of episodes, but an approximation of $Q^*(s, a)$ can be achieved for a given domain by iterating a large enough number of episodes. The number of episodes required for convergence varies significantly from domain to domain, based largely on its size and the nature of the connections between states.

3. Q-LEARNING AND CHORD SEQUENCE GENERATION

In order to generate chord sequences, Reese *et al.* apply the Q-Learning model to an n-dimensional domain composed of a number of states, each of which represents a distinct chord of n pitches from the chromatic (12-note) musical scale [3]. Each state in the domain is equivalent to one chord. The space is toroidal, in that motion in a given direction loops the pitch classes in a manner similar to the circle of fifths (Krumhansl [7] provides a helpful description of chord relations), and chords representing all possible combinations of n pitch classes may be situated in the n-dimensional space.

An example of a two-dimensional representation of a domain containing all possible two-note interval combinations is provided by Tymoczko [8] and reproduced in Fig. 2. Motion along the horizontal axis reflects transposition and movement in the vertical axis increases or decreases the interval size. Reese *et al.* define each chord in the grid produced by Tymoczko as a state, and an agent, starting at one of the chords, can create sequences of chords by performing actions (*i.e.* moving from state to state). Thus the agent's traversal of this domain can be represented as a sequence of two-note musical harmonies. In Fig. 2, arrows indicate one possible pathway through the domain; the chord sequence thus generated is notated in Fig. 3.

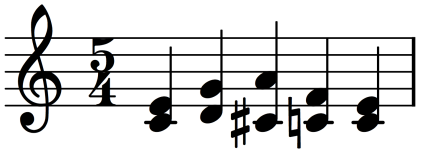

Figure 3. A chord sequence resulting from the harmonic domain traversal shown in Fig. 2

Reese *et al.*'s Q-learning implementation for chord transition composition defines an action as a vector v_{ij} representing the direction and speed of the agent as it moves from one chord state to another chord state. This vector is a representation of the combined motion $P(v_{ij}) = P(v_i)P(v_j)$ of each note in the chord in the pitch space, where v_i describes the direction, and v_j the amount, of movement for each note in a chord transition. If a start position and end position are then defined for the agent in the domain, and the system is allowed to iterate, the result is a tonal chord transition from one state to another. Reese *et*

[3] Although this discussion assumes 12-tone equal tempered pitch classes, this technique could easily be extended to microtonal domains

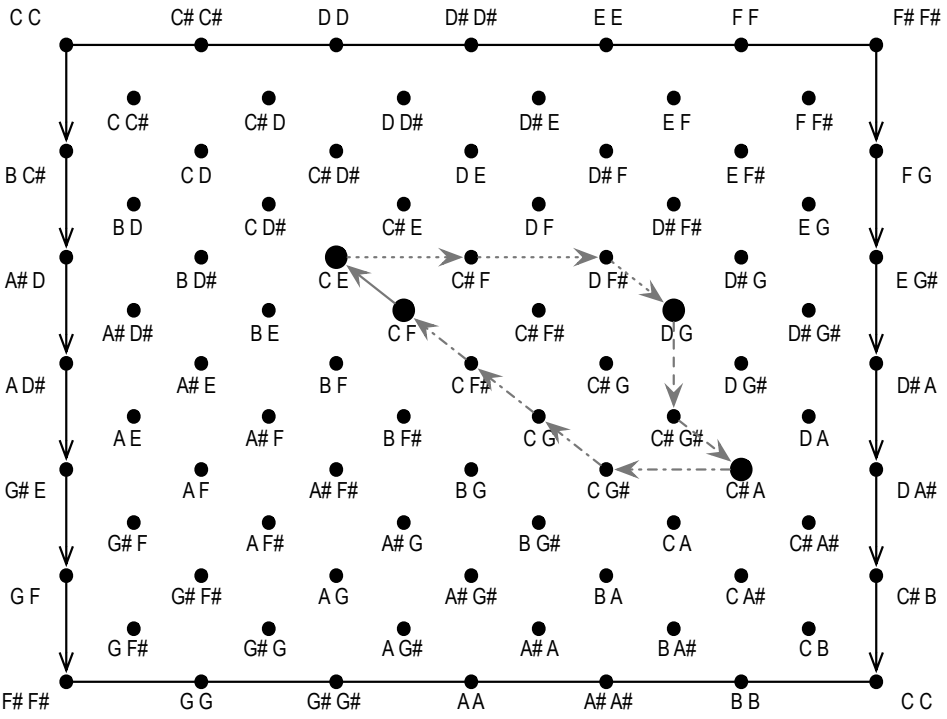

Figure 2. A 2-dimensional harmonic domain (after [1]) showing a path traversal that could be used to generate the sequence in Fig 3.

al. indicate (and admit) that although pathways thus established represent a sequence of chords, they do not necessarily lead to a musically pleasing result. Biasing of the reward matrix is required, and Reese *et al.* performed this biasing using subjective, hand-tuned, trial-and-error reward values.

Reese *et al.*'s Q-learning implementation can generate sequences of chords that transition between any two user-defined chords, and their output (sequences of chords with defined beginning and end chords) demonstrates a tendency to include chords that express a particular tonality. For example, in one generated chord sequence, they observe the consistent appearance of the pitch C in the tenor voice of chord series (where n=4) that begins on a C-Major chord, showing evidence that their system chooses chords based in part on their conformance with a desired tonal centre. The disadvantages of their approach can include excessive chromaticism (use of non-scale notes) in generated chord sequences that reduce the subjective impression of a particular tonality (major or minor), and the relative reduction of efficiency of their algorithm resulting from the wide range of states and transitionary actions that must be accounted for within their domains.

4. NEW APPROACHES TO REAL-TIME SEQUENCE GENERATION

This paper proposes three specific improvements to Reese *et al.*'s model of Q-learning for chord sequence generation:

1. To increase the efficiency of the Q-Learning process by reducing the number of chord states in a domain;
2. To generate more aesthetically pleasing chord sequences by selectively excluding chord states that diverge from the tonal space defined by the start chord or goal chord within a domain;
3. To initiate generation of transitional chord sequences prior to approaching convergence, creating greater variation in generated chord sequences while retaining their tendency to conform with desired tonal qualities.

The intent of developing the system presented in this paper is to refine Reese *et al.*s model by reducing the range of the chord space, allowing the elimination of chord choices that depart too extremely from the tonality defined by the start and goal chords. When a composer creates music, they are generally aware that certain pitches and intervals are more common and useful in a given key than others, particularly when attempting to achieve specific aesthetic goals. Despite biasing the reward matrix with values that strongly promote major (and to a lesser degree, minor) chords, Reese *et al.*'s generated chord sequences still can exhibit a high degree of chromaticism, since all possible chords may be represented as states in their implementation of the framework. A composer writing in a traditional idiom such as classical or traditionally-influenced cinematic film music knows that there are certain chords that are much less commonly seen in a particular style of music (generally producing aesthetically unsatisfying or unpredictable results) and can be ignored for reasons of practicality in many circumstances.

For the interested reader, audio/MIDI samples can be made available that demonstrate the results of the experiments as

described below. To obtain these samples, please email the authors directly.

4.1 Restricting chord states

One of the intents of the system we present is that it be able to generate novel chord sequences in real time. Because of the computationally heavy nature of the system involved, one technique for reducing the time of execution is to reduce the number of possible states, making learning and traversing the Q-matrix quicker. One may suggest that this adds similar constraints to the diversity of chord sequences possible in the system, similar to the result of the hand-tuning of rewards that Reese *et al.* use, however, the restriction in the number of states in no way restricts the order of traversal of the states, and the chord state restrictions can be seen as an *enabling constraint*, allowing new interesting sequences to emerge from what may be seen as relatively simple domains. Further, the restrictions that follow allow us to develop a simplified system that proves the concepts herein. Similar simplifications to the domain could be made with any characteristics of the chords or sequences thus generated.

We propose to restrict the domain of included chords by first assuming that the start and goal chords are either major or minor (chords which are common in the tonal music of film and video games). Since both major and minor chords comprise a subset of the notes within the five-note pentatonic scale (for example, the C-Major chord, with notes C-E-G, represents a subset of the C-Major pentatonic scale, C-D-E-G-A), it is possible to restrict the domain of chord states in which Q-Learning will occur to include only chords that may be built from the pentatonic scales of the start chord, the goal chord, or a combination of the two. An example of such a domain (based on C-Major) is shown in Fig. 4, which shows a complete domain for chords of n=2 (*i.e.* two-note intervals) with excluded chords greyed out.

Since this domain is significantly smaller than the full domain, it can be expected that the agent will, during an episode, generally take fewer steps to reach the goal state. The agent's pathway is randomly directed and the number of choices of action that the agent may have at any given state is reduced, meaning that each step through the domain is more likely to reach the goal state.

A domain based upon the pentatonic scales of the start chord and goal chord is constructed as in Fig 4. Any chord states within this domain consist entirely of pitches that are found in the pentatonic scales based on the start and goal chords. Any chord states that contain pitches not found in either of the two pentatonic scales are excluded. If the start and goal chords are the same, then all chords in the domain will be built exclusively from the five pitches in the pentatonic scale based on that chord. If the pentatonic scales do not overlap at all, then the domain will feature chord states that consist of a set of ten pitches.

Pentatonic scales are used this way because they contain within them a large number of consonant intervals, including perfect fourths, perfect fifths and major thirds. They also exclude several dissonant intervals such as the tritone and minor second. Thus music built out of the notes of a pentatonic scale tends to take on a pleasingly consonant aspect (as the domain of video game music would typically require), and it is intended that the chord transitions generated by this system embody this consonant quality. Similarly, if the compositional domain required dissonant sequences instead of consonant sequences, a set of intervals could be selected to allow these sequences to appear.

4.2 Biasing the reward matrix

Given an appropriate choice of reward structure, the passage can embody a variety of different stylistic and aesthetic aims. Reese *et al.* utilize a reward matrix with a single dimension R(s), in which each destination chord state is associated with a particular reward as described above. In the modified system proposed here, the reward matrix is two dimensional, with the reward for an action being based both on the qualities of the destination chord and its relationship with the origin chord from which the individual action was taken. Using this system, it is possible to encourage motion within a pentatonic tonality by providing larger rewards for actions that remain within a pentatonic tonality.

R	A1 C–D	A2 C–E♭	A3 C–G	A4 E♭–F	A5 A–B♭
S1 C–D		−50	+200	−50	−50
S2 C–E♭	−50		+200	+100	−50
S2 C–G	+100	+100		+100	−50
S4 E♭–F	−50	+100	+200		−50
S5 A–B♭	−50	−50	+50	−50	

Table 4. A subset of the reward matrix representing biases calculated based on the given rules.

Table 4 provides a subset of a reward matrix for a transition from a C-Major chord state to an E♭-Major chord state. The reward matrix punishes actions that involve motion between C-Major and E♭-Major (*e.g.* motion from E♭-F to C-D is punished with a value of -50) and rewards (consonant) motion within a pentatonic scale (*e.g.* E♭-F to C-E♭if rewarded with +100). Between these two scales there is also an interval that is contained within both scales, that being C-G. Because this interval is found in both scales, it can act as a "pivot" point, allowing for transitions between the C-Major tonality and the E♭-Major tonality. Any state/action pair that results in the agent reaching this pivot state is generously rewarded, as can be seen in Table 4.

The goal chord may also be heavily biased against (with a very low reward value), to ensure that motion in chord space avoids this chord. The goal chord may be added to the end of the generated transitional sequence manually after the sequence is generated, and biasing against its appearance in the transitional sequence helps to ensure that the chord states represented in the transition are not repetitive or harmonically redundant. To create the sense of a cadence leading from a generated chord transition to the goal chord, the dominant chord in the key of the goal chord

C–C		C#–C#		D–D		D#–D#		E–E		F–F		F#–F#
	C–C#		C#–D		D–D#		D#–E		E–F		F–F#	
B–C#		C–D		C#–D#		D–E		D#–F		E–F#		F–G
	B–D		C–D#		C#–E		D–F		D#–F#		E–G	
A#–D		B–D#		C–E		C#–F		D–F#		D#–G		E–G#
	A#–D#		B–E		C–F		C#–F#		D–G		D#–G#	
A–D#		A#–E		B–F		C–F#		C#–G		D–G#		D#–A
	A–E		A#–F		B–F#		C–G		C#–G#		D–A	
G#–E		A–F		A#–F#		B–G		C–G#		C#–A		D–A#
	G#–F		A–F#		A#–G		B–G#		C–A		C#–A#	
G–F		G#–F#		A–G		A#–G#		B–A		C–A#		C#–B
	G–F#		G#–G		A–G#		A#–A		B–A#		C–B	
F#–F#		G–G		G#–		A–A		A#–A#		B–B		C–C

Figure 4. An example of a pentatonic chord space built on the pentatonic scale with root "C". Chords that contain pitches not included within this pentatonic scale are greyed out, and would not be included among the states of a domain built upon a C-Major chord.

All Pitches	C	C♯	D	E♭	E	F	F♯	G	A♭	A	B♭	B
A-minor	C		D		E			G		A		
E♭-Major	C			E♭		F		G			B♭	
Exclude		C♯					F♯		A♭			B

Table 3. Exclusion of pitches based upon the start and goal chords. Only chords built entirely from pitches that are represented within the pentatonic scales built from A-minor and E♭-Major are permissible in the domain.

may be inserted immediately before it in the sequence. Because the dominant chord is highly compatible with the tonal space inhabited by the goal chord, its insertion into a sequence as the penultimate chord generally creates a heightened sense that the sequence is approaching the goal chord in a manner consistent with norms of tonal music.

4.3 Variability in generated sequences

If enough episodes are undertaken, a Q Matrix will approach convergence, wherein an optimal path may be identified linking the start state and goal state. In the context of dynamic music, however, it is often advantageous to include some uncertainty in the structure of a chord transition. The video game player might hear a transition between two themes dozens of times, depending on the game and user behaviour, and any randomness in the structure of the transition will help the music to remain fresh and unpredictable. However, it is also desirable for generated chord transition sequences to conform to the tonal norms as already described.

The counterintuitive solution that we propose here is to run the algorithm for only a small number of episodes, such that we deliberately *avoid* convergence. Since Q matrix values are updated based on the random motions of an agent through the chordal domain, lower numbers of iterations will result in Q matrices that are more heavily biased in areas that happened to be more frequently travelled by the agent. The less the agent has travelled through the domain, the more the different regions of the domain will be differentially biased based on the agent's random choices of states to enter. The number of iterations that the agent undertakes becomes a variable that is proportional to the amount of variation exhibited in generated transitions.

4.4 Applications in the context of a video game

To apply the system described here to creating transitional material in a video game, it is necessary for the composer of the video game score to provide a representation of the harmonic map of each cue in a video game. For example, suppose the player is exploring a beautiful environment accompanied by a pastoral musical theme "P", written in C-Major. At some point, the game spawns a monster, necessitating the transition to the energetic melee theme "M", written in E♭-Major. In systems like iMUSE, which involve precomposed music, the transition would have to occur at some marker point in theme P, and theme P would likely have to remain in a more restricted tonal space, never moving too far away from C-Major, if the precomposed transition were to function properly at a variety of different marker points within theme P.

Since the Q-Learning system described here is capable of transitioning from any chord to any other chord, it is possible for transitions to occur from *any point* in theme P, so long as the chord in theme P which begins the transition is known. There is no need for special code to determine this map of chord changes in theme P, since the composer can enter this information manually. A generative transition system such as this can thus support transitions from any point in a theme for which the chord is known, and the cue can range widely through tonal space, since it is not necessary to support transitions that begin in a limited range of chords.

C-Maj	F-Maj	G-Maj	A-min
D-min	E-min	A♭-Maj	B♭-Maj

Table 5. A possible harmonic map of a generic theme.

The theme in Table 5 consists of eight measures each of which express a particular chord as labelled. If theme M is known to begin with the chord E♭-Major, then creating a transition is simply a case of determining which point in theme P the transition will begin from (taken from the map in Table 5) and generating a transition to E♭-Major as described above.

5. CONCLUSIONS

The system proposed here has several advantages over earlier systems:

- Transitions may be generated from any chord to any other chord, meaning that composers are free to range as far as they want in tonal space. Previous systems have restricted the composer to a narrow harmonic area in a single cue.
- There is an inherent randomness in the generated chords, but these chords still conform to a desired tonal space. This quality helps to reduce the risk of repeated transitions becoming stale and predictable, an advantageous quality in a game where specific transitions may appear hundreds of times during a single play-through.
- The reduction in the size of the domain (through excluding undesirable chord states entirely) serves to reduce the computational complexity of the system, increasing its speed and potentially allowing it to be implemented in real time during execution of the game code.
- Biasing the reward table affects the tonal qualities of generated transitions, leading to the possibility of using the reward table as an expressive tool. It is possible, for example, to promote transitions that focus on minor chords by increasing the reward for them in the reward matrix.
- By converting the reward matrix from one dimension to two dimensions, it is possible to base the reward for an action not just on the quality of the destination chord (*e.g.* whether it is major or minor), but on the qualities of the relationship between the current chord state and the destination chord.

Through implementation of this system, it is theoretically possible for composers of adaptive music (such as that of video games) to focus their efforts on composing cues, rather than concerning themselves with composing the potentially large number of required transitions between cues as well. They would be free to write more challenging music since they would not be restricted to confining each music cue to a narrow range of keys; rather, their compositions might move around the circle of fifths quite freely. With creative manipulation of the reward table and choice of the number of episodes to iterate before generating a chord sequence, the composer can create desirable variability in their generated chord transitions while still maintaining conformance with the tonalities of the start and goal chords. As such, this application of the Q-Learning framework serves both as a creative tool and an efficient means of refocussing the adaptive music composer's efforts on thematic material.

6. REFERENCES

[1] K. Reese, R. Yampolskiy, and A. Elmaghraby, "A framework for interactive generation of music for games," *Proceedings of CGAMES'2013 USA*, vol. 0, pp. 131–137, 2012.

[2] J. Hatchman. (2013, November) Know the score: The music of grand theft auto v. [Online]. Available: http://www.clashmusic.com/features/know-the-score-the-music-of-grand-theft-auto-v

[3] K. Collins, *Game Sound: an introduction to the history, theory, and practice of video game music and sound design*. Mit Press, 2008.

[4] C. J. C. H. Watkins, "Learning from delayed rewards." Ph.D. dissertation, University of Cambridge, 1989.

[5] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[6] D. L. Poole and A. K. Mackworth, *Artificial Intelligence: foundations of computational agents*. Cambridge University Press, 2010.

[7] C. L. Krumhansl, "The geometry of musical structure: A brief introduction and history," *Comput. Entertain.*, vol. 3, no. 4, pp. 1–14, Oct. 2005.

[8] D. Tymoczko, "The geometry of musical chords," *Science*, vol. 313, no. 5783, pp. 72–74, 2006.